2

PL-TR-92-2248 (II)

2

ANALYSIS OF DYNAMICAL PLASMA INTERACTIONS WITH HIGH-VOLTAGE SPACECRAFT

M. J. Mandell
T. Luu
J. Lilley

Maxwell Laboratories, Inc.
S-Cubed Division
P. O. Box 1620
La Jolla, California 92038-1620

June 1992

Final Report - Volume II
31 December 1988 through 31 December 1991

PHILLIPS LABORATORY
AIR FORCE MATERIEL COMMAND
HANSCOM AIR FORCE BASE, MA 01731-5000

" This technical report has been reviewed and is approved for publication "

David L Cooke
DAVID L. COOKE
Contract Manager

[illegible]
CHARLES P. PIKE
Branch Chief

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	June 1992	Final (31 Dec. 1988 - 31 Dec 1991)

4. TITLE AND SUBTITLE

Analysis of Dynamical Plasma Interactions With High-Voltage Spacecraft

5. FUNDING NUMBERS

PE 62101F
PR 7601
TA 30
WU CA
Contract F19628-89-C-0032

6. AUTHOR(S)

M. J. Mandell J. Lilley
T. Luu

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Maxwell Laboratories, Inc.
S-Cubed Division
P. O. Box 1620
La Jolla, CA 92038-1620

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Phillips Laboratory
Hanscom Air Force Base, MA 01731-5000

Contract Manager: David Cooke/WSSI

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

PL-TR-92-2248 (II)

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This volume is the users' and programmers' manual for the DynaPAC computer code. DynaPAC is a user-friendly and programmer-friendly workbench for studying plasma interactions with realistic spacecraft in three dimensions. DynaPAC enables plasma interactions specialists to perform predictive scientific calculations with direct application to engineering problems. DynaPAC uses a finite element algorithm with strictly continuous electric fields for accurate potential calculation and particle tracking, and contains a powerful database utility for storage and communication of large data arrays.

The core capabilities of DynaPAC are (1) to define the spacecraft geometry and the structure of the computational space; (2) to solve the electrostatic potential about the object with space charge computed either fully by particles, fully analytically, or in a hybrid manner; and (3) to generate and track representative macroparticles in the computational space. The core modules are designed to have the maximum practical user control and to facilitate the incorporation of new or modified algorithms. Preprocessors are provided to set boundary conditions and generate input files in a modern, screen-oriented way. Similarly, screen- oriented postprocessors are provided for graphical and textual data display

14. SUBJECT TERMS		15. NUMBER OF PAGES
Spacecraft Plasma Interactions	Particle in cell	130
Space Power	Plasma Sheaths	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

Table of Contents

Preface

This is Volume 2 of a two volume final report for the contract "Analysis of Dynamical Plasma Interactions with High Voltage Spacecraft." The period of technical performance was 31 December 1988 through 31 December 1991. The objectives of this contract were to study dynamical plasma interactions with high voltage spacecraft, to construct a three-dimensional computer code, DynaPAC, as a workbench for such studies, and to support the SPEAR program. Volume 1 is a compilation of work done to model high voltage plasma interactions, with application to the SPEAR-II chamber tests and to the design of SPEAR-3. Volume 2 contains documentation for the DynaPAC code, as well as for the two-dimensional Gilbert code.

1. Introduction and Overview

1.1. Need for DynaPAC

At the inception of this contract, recent flight experiments (notably SPEAR-I) and concurrent computer modeling demonstrated a strong capability to predict steady-state interactions between high voltage spacecraft and the space plasma. However, the time-dependent response of the space plasma to the high fields and voltages associated with pulsed power systems, and the associated dynamic spacecraft charging, had not been investigated. Processes not adequately modeled included formation of the space charge sheath, current flow in the quasi-neutral presheath, breakdown phenomena, plasma kinetics, ionization processes, and the effect of dynamic processes on spacecraft charging.

This inadequacy became apparent in trying to make plasma interaction predictions for the SPEAR-II high voltage test. Equilibrium sheath calculations gave very different results from sheath calculations using short time approximations, and plasma currents to the high voltage components could not be calculated with confidence. Thus, it was decided to employ S-CUBED's experience in developing equilibrium computer models to develop a realistic dynamic computational capability incorporating modern and advanced computational techniques.

S-CUBED's three-dimensional equilibrium codes, NASCAP/GEO, NASCAP/LEO, and POLAR, have proved their general utility through application to the design and analysis of a succession of laboratory experiments, spaceflight experiments, and functional spacecraft. The goal of DynaPAC is to provide a similarly general tool which will be used for design and analysis of future pulsed power systems in space and dynamic plasma interactions of other future space systems.

1.2. DynaPAC Capabilities

DynaPAC is currently a user-friendly and programmer-friendly workbench for studying plasma interactions with realistic spacecraft in three dimensions. Presently, DynaPAC enables plasma interactions specialists to perform realistic analyses with direct application to engineering problems. Its current capability is illustrated by its application to SPEAR-II. When mature, DynaPAC will be usable by spacecraft engineers with plasma interactions expertise.

The core capabilities of DynaPAC are to

(1) Define the spacecraft geometry and the structure of the computational space;

(2) Solve the electrostatic potential about the object, with flexible boundary conditions on the object and with space charge computed either fully by particles, fully analytically, or in a hybrid manner; and

(3) Generate, track, and otherwise process representative macroparticles of various species in the computational space.

The core modules are designed to have the maximum practical user control and to facilitate the incorporation of new or modified algorithms. Preprocessors are provided to set boundary conditions and generate input files in a modern, screen-oriented way. Similarly, screen-oriented postprocessors are provided for graphical and textual data display.

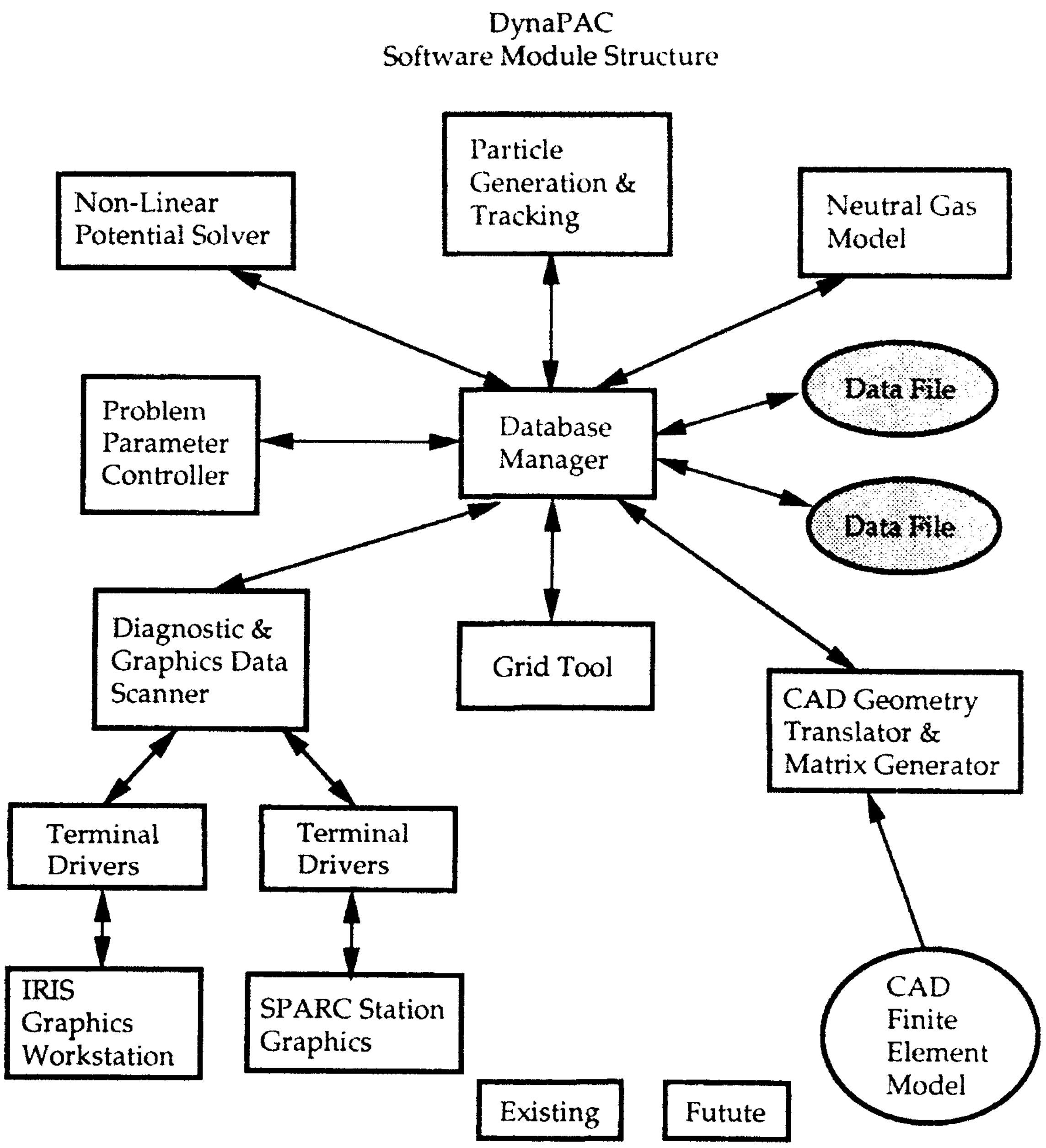

DynaPAC
Software Module Structure
Non-Linear Potential Solver
Particle Generation & Tracking
Neutral Gas Model
Problem Parameter Controller
Database Manager
Data File
Data File
Diagnostic & Graphics Data Scanner
Grid Tool
CAD Geometry Translator & Matrix Generator
Terminal Drivers
Terminal Drivers
IRIS Graphics Workstation
SPARC Station Graphics
CAD Finite Element Model
Existing
Futute

Figure 1.1. Modular structure of the DynaPAC code.

1.3. DynaPAC Software Elements

Figure 1.1 shows the modular structure of the DynaPAC code. Except for the "Neutral Gas Model," all modules shown are currently operational.

DynaPAC is centered around a DataBase Manager. The DataBase Manager is a library of routines capable of making large arrays of information contained in disk files accessible to computational modules. It has a programmer-friendly language for defining data types and for retrieving and storing data. The DataBase Manager will be used to write DynaPAC results in the format required for other application codes, and to retrieve information generated by other codes for use in DynaPAC. The DataBase strategy enables DynaPAC to be operable on, and portable among, modern high-power workstations, which have proven to be more cost-effective than supercomputers for this type of code development and analysis.

Spacecraft geometrical definitions for DynaPAC are done using standard finite element pre-processors such as PATRAN. Among the advantages of this approach are that the geometry can be realistically represented, and that finite element models of a spacecraft constructed for other purposes can be adapted for DynaPAC use. The computational space around the spacecraft is constructed interactively using the GridTool module. Arbitrarily nested subdivision allows resolution of important object features while including a large amount of space around the spacecraft. The DynaPAC object definition interface program (*e.g.*, PatDyn) initializes the DataBase and constructs the properties of those finite elements neighboring the spacecraft.

DynaPAC uses a high-order finite element representation for the electrostatic potential that assures that electric fields are strictly continuous throughout space. (The originally proposed tri-quadratic finite element representation proved to have some unfortunate properties, and was replaced by the new scheme.) The electrostatic potential solver uses a conjugate gradient technique to solve for the potentials and fields on the spacecraft surface and through the surrounding space. Space charge options presently built in include Laplacian, equilibrium sheath, "frozen ions" (appropriate to the early stage of a negative transient pulse), "mobile ions - barometric electrons" (appropriate to the several microsecond timescale response to a negative pulse), and "full PIC". A screen-oriented, menu-driven preprocessor is available to generate initial conditions and to build an input file for the potential solver.

Particle tracking is used to study sheath currents, to study particle trajectories, or to generate space charge evolution for dynamic calculations. DynaPAC generates macroparticles either at a "sheath boundary" or throughout all space. Particles are tracked for a specified amount of time, with the timestep automatically subdivided at each step of each particle to maintain accuracy. The current to each surface cell of the spacecraft is recorded for further processing. A future task is to develop algorithms which will allow DynaPAC to study problems having neutral ionization.

A screen-oriented, menu-driven postprocessor is used to access the DataBase files and generate graphical output illustrating such quantities as object surface potentials, space potentials, particle positions, or particle trajectories. Potential contour plots may be made either as line plots or as color-filled plots. Contour levels, plot range, *etc.* can be modified through the user interface. Plots are written to an intermediate graphics file which can be displayed using any of several terminal protocols or converted to PostScript files for printing. This scheme is designed to easily accommodate new graphics hardware or software.

2. DynaPAC Users Manual

2.1. Preparing to Use DynaPAC

This manual describes the use of DynaPAC from the user's point of view. Because DynaPAC is a large set of complex codes, tools have been developed to make file handling and input generation as automated an well-interfaced as possible.

DynaPAC should be installed on a reasonably powerful UNIX workstation. The DynaPAC program tree requires about 50 MB of disk space. At least several MB of disk space should be available for problem files. Complex problem file sets can easily require 100 MB or more.

2.1.1. Units

In general, DynaPAC operates in the SI or MKS system of units. Electrostatic potentials are internally stored in volts, and electric fields in volts per meter. Magnetic fields are always in tesla (or webers per square meter). Particle energy or plasma temperature is usually in electron-volts. Charge density (ρ) is usually divided by the permittivity of free space (ε_0), so that it has the units of volts per square meter.

2.1.2. DynaPAC Environment

DynaPAC makes use of UNIX environment variables and aliases. These are defined by the "DynaPAC.setup" file, found in the head of the DynaPAC tree. Figure 2.1 shows the "DynaPAC.setup" file. Only the first three "setenv" commands need to be customized to your system: the SCREENPKG variable needs to be set to the head directory of the DynaPAC screen package, the DYNAPAC variable needs to be set to the head directory of DynaPAC, and the CPU_TYPE variable needs to be set to either IRIX (for Silicon Graphics, Inc. computers with IRIX 4.0 or later operating system) or SUN4 (for Sun Microsystems Sun-4 or SparcStation computers).

DynaPAC executables are suffixed with the CPU_TYPE variable. Those executables with screen interfaces must be run using shell scripts which set additional environment variables prior to running the executable. The aliases in the "DynaPAC.setup" file make all this transparent to the user. It is not necessary (or recommended) that the $DYNAPAC/bin directory be included in the user's "path". It is recommended that the DYNAPAC variable be defined in the user's ".cshrc" file.

```
#
# %W% %G% S-CUBED

# setup new dynapac window
# check/update the shell variables below

# shell variables
#   DYNAPAC   is the head of Dynapac node
#   SCREENPKG is the head of ScreenPkg
#   CPU_TYPE  is machine type
#             (SUN4, SUN3, R3200, IRIS, IRIX, STELLIX)
setenv SCREENPKG /itch/ScreenPkg
setenv DYNAPAC   /itch/dynapac
setenv CPU_TYPE  IRIX

# screen package things
setenv TERMCAP    $SCREENPKG/src/termcap/termcap
setenv FORMGENDIR $SCREENPKG/bin

# aliases
alias formgen $FORMGENDIR/formgen
alias Dmake   $DYNAPAC/dynamake

# executables
alias Potent  $DYNAPAC/bin/Potent_$CPU_TYPE
alias PatDyn  $DYNAPAC/bin/PatDyn_$CPU_TYPE
alias PolDyn  $DYNAPAC/bin/PolDyn_$CPU_TYPE
alias NisDyn  $DYNAPAC/bin/NisDyn_$CPU_TYPE
alias PartGen $DYNAPAC/bin/PartGen_$CPU_TYPE
alias Tracker $DYNAPAC/bin/Tracker_$CPU_TYPE
alias ObjPotl $DYNAPAC/bin/ObjPotl_$CPU_TYPE
alias MakeNpl $DYNAPAC/bin/MakeNpl_$CPU_TYPE
alias AddGrid $DYNAPAC/bin/AddGrid_$CPU_TYPE

# executables with screen interface
alias DynaPost $DYNAPAC/bin/DynaPost
alias DynaPre  $DYNAPAC/bin/DynaPre
alias GridTool $DYNAPAC/bin/GridTool
alias Scanner  $DYNAPAC/bin/Scanner

# drivers
alias dbtool $DYNAPAC/src/dblib/dbtool

# greetings
echo ' '
echo Welcome to DynaPAC ...
echo ' '
cd $DYNAPAC
pwd
```

Figure 2.1. "DynaPAC.setup" file.

To set up the aliases and environment variables needed to run DynaPAC, change directories to the DynaPAC head directory and type

```
source DynaPAC.setup
```

or (from any directory)

```
source $DYNAPAC/DynaPAC.setup
```

You will be left in the DynaPAC head directory.

2.1.3. DynaPAC Modules

DynaPAC is a collection of executable modules which perform the core functions of DynaPAC and associated pre- and post-processing. The first module that a user will encounter is not part of DynaPAC at all, but an external finite element geometry generator (presently either PATRAN or POLAR) used to define a geometrical object for DynaPAC simulation. Then DynaPAC modules will be applied to the object in more or less the following order:

GridTool is used to interactively define an arbitrarily nested cubic grid system for the space surrounding the object.

DynaPre is used to (1) define input for the geometry interface modules, **PatDyn** and **PolDyn**. DynaPre will later be revisited to (2) define initial potential boundary conditions on the object surfaces; (3) define input to the potential solver module, **Potent**; (4) define input to the particle generator module, **PartGen**; and (5) define input to the particle tracker module, **Tracker**.

PatDyn reads a PATRAN neutral file and generates the surface and volume information required by the DynaPAC computational modules.

PolDyn reads geometrical information from POLAR program files and generates the surface and volume information required by the DynaPAC computational modules.

Potent is used to calculate the electrostatic potential field in the space surrounding the object.

Scanner is primarily used to display the electrostatic potential field in the space surrounding the object. It can also be used to display selected other spatial variables, and to print any valid DynaPAC spatial or surface variable.

PartGen is used to generate macroparticles to study particle trajectories, surface currents, and space charge.

Tracker is used to track macroparticles to study particle trajectories, surface currents, and space charge.

DynaPost is a postprocessor to display (and plot time history of) values of surface currents, and to generate input for (and run) the ObjPotl module.

ObjPotl plots isometric views of the object surface, color-coded by material number, conductor number, surface potential, surface electric field, or incident flux.

2.1.4. DynaPAC Files

The user brings to DynaPAC's GridTool module files describing the geometric and material configuration of his or her object. At that time a **prefix** will be assigned to the file set. From that point, all files are created by the DynaPAC modules. In general, the following naming conventions are followed:

prefix.suffix files (where **suffix** is not capitalized) are GridTool ascii output files. They may be edited by the user (though altering them is not recommended).

prefix.SUFFIX files (where **SUFFIX** is capitalized) are DataBase Manager files. In general, these are non-ascii and cannot be user modified. Each such file has a maximum size of 33.5 MB.

name_in files (where **name** identifies a module) are ascii input files. They contain well-labeled parameters whose values may be altered by the user.

name_out files (where **name** identifies a module) are output files. These consist primarily of reports of the progress of the module, but also contain information of interest to the analyst. Various modules also write auxiliary output files.

Scratch.XX files are created by the Potent module and may be deleted any time Potent is not operating.

fort.2 files are graphics files, and are overwritten each time a figure is generated by a DynaPAC module. Figures can be saved by renaming the corresponding fort.2 file.

More specifically, here is a list of some important DynaPAC files with their contents:

prefix.neu is a PATRAN neutral file (ascii) containing object geometrical information.

prefix.mat is a user-generated file (ascii) containing material definitions.

prefix.grd (ascii) is generated by GridTool to define the DynaPAC grid system for the problem.

prefix.obj (ascii) is generated by GridTool as a simplified object description.

prefix.DP is the main DataBase file for the problem.

prefix.HI (ascii) contains the data packet names and definitions for the DataBase Manager.

prefix.BS and **prefix.MEnn** are generated by PatDyn or PolDyn to contain "special" element information. These files are treated as read-only by subsequent modules, and can be shared among different versions of a problem using the same geometry and gridding. Also, the **prefix.MEnn** files are read only by the Potent module, and may be deleted if no further potential solutions are anticipated.

prefix.PTn files are used to store current macroparticle information.

prefix.POTG, **prefix.POTS**, and **prefix.CUR** store time histories of spatial potentials, surface potentials, and surface currents.

User-modifiable input files include

Default Name	Module	Generated By
caddyn_in	PatDyn	DynaPre
caddyn_in	PolDyn	DynaPre
ps_in	Potent	DynaPre
pg_in	PartGen	DynaPre
tr_in	Tracker	DynaPre
tr_traj_in	Tracker	DynaPre
objpotl_in	ObjPotl	DynaPost

Names for standard output files are defined by the user.

2.2. Handling Input and Output

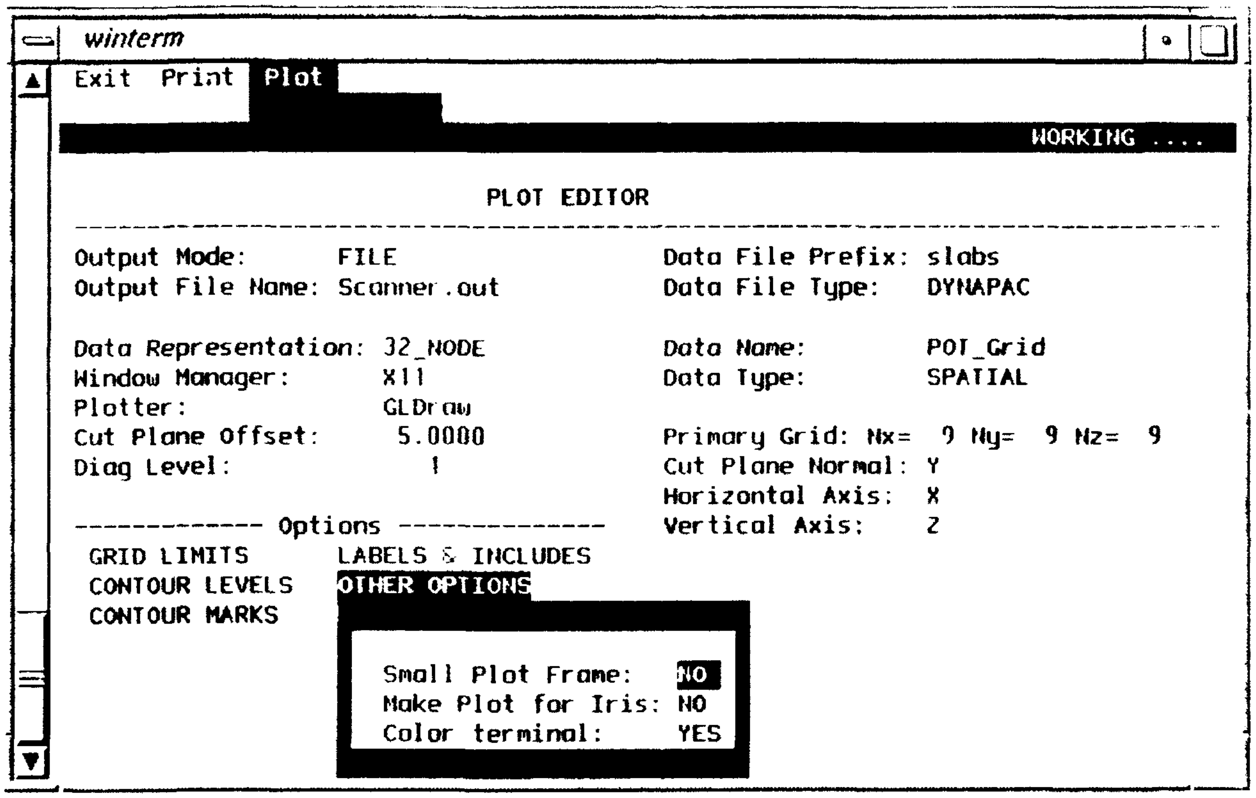
The modules of DynaPAC are conveniently divided into computational modules (such as PatDyn, Potent, PartGen, Tracker, and ObjPotl), and interactive modules (such as GridTool, DynaPre, Scanner, and DynaPost). The computational modules read their parameters and directives from standard input and write (mostly diagnostic) information to standard output. For these modules it is convenient to obtain an initial input stream (from DynaPre or DynaPost) and use the UNIX editor for subsequent modificatio.1s of parameter values. The interactive modules obtain user input through the screen handler, and produce output via screen display, graphics files, or ascii files. We assume the user is familiar with the standard UNIX utilities, and describe here how to use the DynaPAC Screen Handler and the graphics display interfaces.

2.2.1. Using the Screen Handler

The screen handler provides a method of displaying parameter values to the user in a convenient and well organized form, while allowing him or her to alter the values directly on the form. The scheme works on terminals, terminal windows, or terminal emulators

2.2.1.1. Appearance of the Screen

Figure 2.2 shows a typical screen. On the top line appears the static top level menu for the module being run. Below that (in reverse video where available) is the "annunciator line", which occasionally contains useful messages about the process being performed. Second and third level menus and screens are superimposed on one another, with only the lowest level menu accessible to the user. Menus and screens vary in size from small pulldowns to the entire area below the annunciator bar, depending on the space required for their function.



```
                         PLOT EDITOR
-----------------------------------------------------------------------------
Output Mode:       FILE                 Data File Prefix: slabs
Output File Name: Scanner.out           Data File Type:   DYNAPAC

Data Representation: 32_NODE            Data Name:        POT_Grid
Window Manager:      X11                Data Type:        SPATIAL
Plotter:             GLDraw
Cut Plane Offset:     5.0000            Primary Grid: Nx=  9 Ny=  9 Nz=  9
Diag Level:             1               Cut Plane Normal: Y
                                        Horizontal Axis:  X
------------- Options --------------    Vertical Axis:    Z
 GRID LIMITS       LABELS & INCLUDES
 CONTOUR LEVELS   OTHER OPTIONS
 CONTOUR MARKS

                     Small Plot Frame:   NO
                     Make Plot for Iris: NO
                     Color terminal:     YES
```

Figure 2.2. Example of DynaPAC screen.

2.2.1.2. Types of Fields (with Examples)

Several types of fields appear on the screens:

Static Fields are used to describe or illuminate the contents of the screen, and are not selectable by the user. For example, the maximum grid dimensions are displayed in the "Plot_Edit - Grid Limits" submenu for user convenience, and may not be changed.

Value Fields contain numerical or ascii values. The values may be modified by positioning the highlight over the value field, selecting the field (with a <CR>), typing the new value, and affirming the entry (with a second <CR>). The screen handler performs some validity checking on value field entries. Examples of value fields are "Plasma Density" and "Plot Title."

Toggle Fields are value fields with a small number of predefined valid entries. Toggle fields are modified by selecting the field (as above), using the spacebar to "toggle" through the possible options, and selecting the desired option (with a <CR>). An example of a toggle field is "UNITS", which may take the values "METERS", "CENTIMETERS", "INCHES", "FEET", or "OTHER". (In this case, choosing "OTHER" results in display of a value field to enter the desired unit in meters.)

Menu Fields result in display of a lower level menu. Selecting the "Edit Script" field results in display of the input parameters for a computational module for user inspection and modification. Similarly, selecting the "Edit Plot" field allows the user to choose the parameters of a plot.

Action Fields result in some action taking place. "Read" causes previous relevant parameter or option values to be read from DynaPAC files. "Write" causes the results of an interactive session to be written. "Make Script" or "Make Plot" cause an input stream or plot to be generated and written on the appropriate file. "Show Plot" causes spawning of a shell to run the currently selected graphical interface program. "Exit" results in escape to the parent menu, and "Help" sometimes gets display of a rudimentary help file. Action fields of the form "Run Program" generally do not work and should be avoided.

2.2.1.3. Navigating the Screen and Menu Tree

The screens and the menu tree are navigated using the letter keys "h", "j", "k", and "l", the Enter <CR> key, and the Escape <ESC> key.

The letter keys move the highlight respectively left, down, up, and right on the current screen or menu. In general, if the requested motion is not possible the screen handler will perform an alternate motion (e.g., up instead of left) or will wrap around.

The Enter <CR> key is used to select a field or to affirm the modified value of a previously selected field.

The Escape <ESC> key is used to restore the previous value of a currently selected field, or (if no field is selected) to signify completion of action on the current menu or screen. Escape from a menu is usually to the parent menu, but may result in display of another menu at the same or a lower level if the context is appropriate. Some menus will not honor the <ESC> key, but require selection of the "Exit" field. In particular, the top level menu does not accept the <ESC> request.

Key	Action
h	Move left (or up)
j	Move down (or right)
k	Move up (or left)
l	Move right (or down)
<CR>	Select field (for modification or action)
<CR>	Accept new value for selected field
<ESC>	Restore old value for selected f. ıd
<ESC>	Signify menu completion

2.2.2. Displaying Graphics

DynaPAC modules write graphics commands to the file "fort.2", from which they may be read and executed by any of several graphical interface programs. The graphical interface programs may be executed by the "Show Plot" menu selection from the interactive modules, or directly from the UNIX shell. In the former case, the correspondence between plot interface name and executable interface is defined by the $DYNAPAC/bin/define_plot_readers file. Note that the "Make Plot" selection appends a plot to the "fort.2" file, and the "Show Plot" selection resets the file pointer to the beginning of the file. Thus, the first "Make Plot" selection following a "Show Plot" selection effectively erases all previous plots.

Currently favored graphical interface programs are listed below. Where not otherwise specified, frame advance is accomplished by a <CR>.

GLDraw or **IRIX_Plot** displays plots through calls to the SGI GL library. The plot window may be moved or resized (with rescaling taking effect on the following frame). Frame advance uses the right mouse button.

XDraw displays plots through calls to the XLib and XToolkit. The plot window may be moved or resized (with rescaling taking effect on the following frame). Frame advance uses any mouse button.

Tek4xxx (where 4xxx is 4014, 4105, 4107, 4207) sends to standard output graphics commands appropriate to the specified Tektronix terminal. The frame advance command is <CR>.

PostGray and **PostColor** generate PostScript commands on standard output. The former replaces the color scale for contour levels with a monotonic gray scale. (No pause for frame advance.)

Adobe generates (on standard output) commands which can be made acceptable for input to the Adobe Illustrator program. (No pause for frame advance.)

SunCore and **SunPix** display graphics commands in the SunView windowing environment. The frame advance command is <CR>.

2.3. Defining Objects and Grids

The first two tasks of a DynaPAC simulation are to define a spacecraft or test object as a boundary surface representation, and to define a grid structure about the object. DynaPAC provides no object definition facility, but accepts definitions produced by other codes. Presently supported are PATRAN (a commercially available finite element analysis code) and POLAR (a spacecraft charging analysis code developed by S-CUBED for the Air Force). Other geometry definition codes may be supported as requirements dictate.

DynaPAC's GridTool module is used to define an arbitrarily nested grid structure about the object. With the definitions of object and grid established, the PatDyn (for PATRAN objects) or PolDyn (for POLAR objects) module is run to analyze the geometrical configuration and establish the DynaPAC database.

2.3.1. Defining Objects using PATRAN

Object definition for DynaPAC is identical to object definition for NASCAP/LEO. For tutorial exercises on using PATRAN to define objects, the reader is referred to the *User's Guide to NASCAP/LEO*. Here we give only the specific requirements that the object must satisfy.

The user must construct the object surface using linear triangles (TRI/3/icond) and linear quadrilaterals (QUAD/4/icond). The PATRAN "configuration code" is interpreted as the "conductor number"; the PATRAN MID is used as the material number. The object should have no free boundaries, and all surface normals should be consistently outward. Nodes should be equivalenced as appropriate, and "COMPACT" and "ELEM COMPACT" operations should be performed prior to writing the neutral file.

The PATRAN neutral file (output as "patran.out.*n*") should be renamed to "*prefix*.neu". A material file which associates material names and properties with the material numbers assigned using PATRAN must appear as "*prefix*.mat". The material file contains two types of packets:

(1) A one-line packet consisting of an integer (MID) followed by a material name. PatDyn then associates the material name with all surfaces have the MID. If the material name is defined in PatDyn's material database, its default properties are assigned. If not, a warning is printed.

(2) A four-line packet consisting of a material name (which must have already been assigned an MID) followed by three lines of material properties in NASCAP/LEO format.

Figure 2.3 shows an example material file.

```
1 Alum
2 Kapton
```

```
3 Teflon
4 NPaint
5 Silver
6 CPaint
7 Gold
8 Aquadg
Gold
     1.00,.001,-1.,79.,.88,.8,88.79,.92
     53.48,1.73,.413,135.,.000029,-1.,1.E+4,2.E+3
     1.E-13,1.,1.E+3,20.
```

Figure 2.3. Example material file, using 8 pre-defined material names, then redefining the properties of "Gold."

2.3.2. Defining Objects using POLAR

It is simpler, although more restrictive, to use POLAR to define DynaPAC objects. The procedure is:

(1) Define a valid POLAR object using POLAR's VEHICL module. Be sure that all POLAR checks are passed. Be sure to correctly define the DXMESH parameter.

(2) Keep POLAR output files "fort.21", "fort.17", and "fort.11". Rename "fort.21" to "fort.8" (mv fort.21 fort.8). Make extra links to these files to avoid their being deleted.

(3) No extra material file is needed. The material names and definitions will be read directly from the POLAR files.

(4) Run GridTool and PolDyn as described below.

2.3.3. Creating Grids using GridTool

2.3.3.1. General

GridTool allows the user to define a grid structure for an object created by PATRAN or POLAR. It runs in a "screen oriented input" mode on UNIX machines with standard ASCII terminals or emulators, and displays graphics using any of the plot display interface modules. A Silicon Graphics Iris version with interactive graphics was developed for an old SGI configuration, but has not been updated to the current SGI windowing system.

Many of the forms or screens shown in this file are slightly out of date, and, in particular, the graphical options do not appear. The differences in the current forms should be self-explanatory.

2.3.3.2. I/O Files

GridTool reads the neutral file (suffix .neu) (renamed from "patran.out.n") from PATRAN, or the files fort.11, fort.17, and fort.8 (renamed from fort.21) from POLAR It creates two output files (suffices .grd and .obj). The output files are also read in for a restart run.

File	Contents
<prefix>.neu	neutral file from PATRAN
fort.11	POLAR file
fort.17	POLAR file
fort.8	POLAR file fort.21
<prefix>.grd	grid parameters (ASCII text)
<prefix>.obj	surface information (ASCII text)

The user will specify <prefix> which will be the same for all three files. The prefix.grd file will be read by PatDyn or PolDyn to define the grid structure. The prefix.obj file is used only for restarts of GridTool.

2.3.3.3. GridTool Capabilities

GridTool presently has the following capabilities:

1. Create and modify a primary grid around a PATRAN or POLAR object.
2. Add and modify a child grid.
3. Delete a grid along with all of its child grids.
4. Restart from an existing grid structure.
5. Graphically or tabularly display the current grid structure.

2.3.3.4. Running GridTool

It is best to illustrate how to use GridTool by going through an example. Let's say we have a neutral file generated by PATRAN, and the file name is 'Cube.neu'. Now, start up GridTool. The first screen will be displayed with the following menu:

Exit	Files	Edit	Show	Write	Reset	Help

Should the 'Help' field be selected, the following screen will be displayed:

Grid Tool Help		
Exit	Leaving GridTool	
Files	Set file prefix	
Show	Show the grids defined so far	
Edit	Edit grids structure	
	Show	show parameters of current grid
	Add	add new child grid to the current grid
	Delete	delete current grid and its children (if any)
	Modify	modify parameters of current grid
Write	Write current grids to file <Prefix>.grd	
Reset	Reset everything to defaults (to start all over)	
Help	This screen	

Normally, the sequence of commands selected are: (1) Files; (2) Edit; (3) Show; (4) Write; (5) Exit. Let's now go through these steps.

2.3.3.4.1. Files

Select 'Files'. The default prefix, 'fort' is shown. Hit <RET> to change it, and type in the correct file prefix which, in this case, is 'Cube'. Then hit <ESC> to exit this menu. A new form with two toggle fields is then popped up:

Neutral file type:	PATRAN
Units used:	meters

Let's say the units used to define the object was in inches. We then would move to field saying 'meters' and to change it to 'inches'. Again, hit <ESC> to exit this form.

2.3.3.4.2. Edit

Select 'Edit'. The default current grid (parent grid) is grid #1. Since we have not defined any grid yet, start from grid #1 is a must. Hit <ESC> to exit this form. We will then be in the following pop-up menu:

Exit
Show
Add
Delete
Modify

Here

Exit	will exit this menu (same as hitting <ESC>)
Show	will show current grid parameters
Add	will add a new child grid to current grid
Delete	will delete the current grid and its children (cannot delete primary grid)
Modify	will modify the current grid (can only modify empty (childless) grids)

Let's select 'Show' to see the current grid parameters. The following form will be shown:

Current grid #	1
Mesh Size (m)	0.0254
Dimension	5 5 5
Child Grid #	

Hit <ESC> to exit this form, then select 'Modify' to make some changes. We will see the following Modify form:

Modifying PRIMARY grid	
Grid dimension:	5 5 5
Mesh size:	0.0254
Object extent in grid units:	
2.5000 <= X <= 3.5000	
2.5000 <= Y <= 3.5000	
2.5000 <= Z <= 3.5000	
TRANSLATE OBJECT	
ACCEPT CHANGES	
IGNORE CHANGES	

We now can move around the form making changes. Say we want to change grid dimensions to 10x10x20, mesh size to 20cm. The form now shows new extents of the object within the grid. Let's also say we want to translate the object 8 mesh units in the -Z direction. Select 'TRANSLATE OBJECT', change Dz to -8, hit <ESC>. Finally, Select 'ACCEPT CHANGES' to save the changes made (or select 'IGNORE CHANGES' to restore previous parameters of this grid.)

Enough for modifying grids, now let's add a child grid. Select 'ADD' to see the ADD form:

New child grid #	2
Parent grid dimension:	10 10 20
Grid limits on:	
(Parent Grid)	(Primary Grid)
5 <= X <= 7	5.0000 <= X <= 7.0000
5 <= Y <= 7	5.0000 <= Y <= 7.0000
10 <= Z <= 12	10.0000 <= Z <= 12.0000
Subdivision Ratio:	2
RECALCULATE	
CHECK OVERLAPS	
ACCEPT GRID	
IGNORE GRID	

A set of default parameters have been set to this newly created child grid. Some parameters of parent grid are included in the form to help us figuring out the location of the child grid. Again, we can move around the form making changes. Other functions include:

RECALCULATE	update the form.
CHECK OVERLAPS	check overlaps between child grids (not relevant here because we only have 1 child grid).
ACCEPT GRID	save this child grid.
IGNORE GRID	ignore this child grid.

Let's say we are satisfied with 2 grids. Select 'Exit' to exit.

2.3.3.4.3. Show

The SHOW option allows either tabular or graphical display of the currently defined grids. In this case, we will illustrate the TABLE option.

Select 'Show'. Then select TABLE from the pulldown menu. The default is to show parameters of all defined grids, but we may choose to look at some subset as well. Exit current form to see the defined grids:

There are 2 grids defined		
Grid #	1	2
Nx	10	7
Ny	10	7
Nz	20	7
Parent	0	1
Mesh Ratio	1	3
Mesh Size	0.02000	0.00667
Limits(in parent grid)		
Range X	1, 10	5, 7
Range Y	1, 10	5, 7
Range Z	1, 20	10, 12

We could go back to 'Edit' again to make changes if we are not pleased with the grids shown. Otherwise, select 'Write' to write new grid structure to file. This will result in 2 files being created: Cube.grd and Cube.obj.

2.3.3.4.4. Exit

Select 'Exit' to exit GridTool. Now, we have three files:

Cube.neu	neutral file	(input)
Cube.grd	grid parameters file	(restart file)
Cube.obj	surfaces data file	(restart file)

The Cube.grd will be:

```
2
1  10  10  20  0  1  10  1  10  1  20
 1.00000  10.00000  1.00000  10.00000  1.00000  20.0000  2.00000E-02
1  1
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
2  7  7  7  1  5  7  5  7  10  12
 5.00000  7.00000  5.00000  7.00000  10.00000  12.0000  6.66667E-03
3  3
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
 2.54000E-02  2.54000E-02  2.54000E-02 0. 0. -0.160000  2.54000E-02
```

The interpretation of the prefix.grd file is:

Line(s)	Field(s)	Contents
1		Total number of grids defined
2-5		Grid 1 parameters
2	1	Grid Number
2	2-4	nx, ny, nz
2	5	parent grid number
2	6-11	limits in PARENT grid coordinates
3	1-6	limits in PRIMARY grid coordinates
3	7	mesh size [meters]
4	1	mesh ratio with respect to PARENT grid
4	2	mesh ratio with respect to PRIMARY grid
5		reserved for future use
6-9		Grid 2 parameters (as 2-5)
10	1-3	Object size in meters
10	4-6	Object center relative to primary grid center [meters]
10	7	Unit conversion factor

2.3.3.5. Afterthought Object Translation

A common occurrence is that, after a grid structure is mostly or fully defined, the user will decide that a grid translation is needed. This can be done in the files prefix.grd and prefix.obj:

1. Write out the grid structure and exit GridTool.
2. Modify fields 4-6 of the last line of the prefix.grd file to reflect the desired object translation [meters].
3. Modify fields 4-6 of the second line of the prefix.obj file exactly as in step (2).
4. Restart GridTool to display and check the new object position.

2.3.4. Processing Objects with PatDyn or PolDyn

PatDyn or PolDyn should be run in a directory containing only the output files from the object definition program and from GridTool, plus a small input file (which can be generated with the CadDyn option of the DynaPre module). Normally, only the PREFIX card is needed. The other options are for diagnostic purposes.

The input options to PatDyn and PolDyn are:

COMMENT

Ignore this card.

DIAGNOSTICS

Print (a large amount of) diagnostic output.

ECHO

Echo the neutral file.

END

Conclude option input.

HELP

Print the available options.

NOECHO

Do not echo neutral file. (This is the default.)

NOPROCESS

Stop before generating any matrix elements.

OFFDIAG

Suppress off-diagonal elements between surfaces.

PREFIX prefix

Define the file prefix for this run.

PROCESS ix jy kz GRID igrid

Proceed as expeditiously as possible to process the requested special element (for diagnostic purposes).

REMARK

Ignore this card.

PatDyn or PolDyn will write an output file of diagnostic information which you may use to monitor its progress. It will initialize the DataBase files, and create the prefix.DP, prefix.HI, prefix.BS, and prefix.MEnn (one for each grid with special elements) files.

Early in execution of the object definition interface file, an important quantity to monitor is "NSpec." (You may "grep" for it.) This is the initially determined number of special volume elements. The current estimate is printed three times for each grid. The limit on special volume elements is 4095.

Many of the "error encountered" messages correspond to cells originally thought to be special, but on careful examination turned out to be full or empty. This message is particularly likely when an object surface is coincident with a grid plane. The final determination (of special, filled, or empty) is usually correct.

2.4. Calculating Potentials

The Potent module of DynaPAC is used to calculate potentials in the space around the spacecraft. Normally, all surface potentials are held at fixed potential. However, it is possible to specify fixed electric field at insulating surfaces, or to allow spacecraft potentials to change in response to plasma currents. Several ways of dealing with space charge are provided, and advanced users are encouraged to extend these capabilities.

2.4.1. Using DynaPre [IPS] to Specify Initial Potentials

When the IPS option is chosen from the DynaPre top menu bar, the following options are presented:

Exit

Return to the DynaPre top menu bar.

Read

Read initial potentials previously written to the DynaPAC DataBase files.

Write

Write initial potentials into the DynaPAC DataBase files. This option must be chosen prior to exiting IPS, or the new specification will not take effect.

Conductors

Specify initial potentials for conductors.

Insulators

Specify initial potentials for insulators.

Show Table

Show the initial potential table.

Reset

Set all cells to fixed potentials of zero volts.

Help

No help is currently implemented for IPS.

The most common initial condition setting is to set all conductors to fixed potentials. This is done with IPS as follows:

(1) Choose "Conductors" from the pull-down menu. Select with <CR>.

(2) Choose "Set" from the "Set/Adjust" menu. Select with <CR>. (The "Adjust" selection is used, for example, when the charge on a system of biased conductors is to be initialized.)

(3) The conductor potential screen now appears as shown in figure 2.4. The "BC Type" column is a toggle between the values "Fixed Potential" and "Biased Potential". If you wish a fixed potential value, choose "Fixed Potential" and enter the value in the third column. If you wish to bias the conductor relative to

another, choose "Biased Potential" and enter the relative value and the conductor number to which it is biased. (The potential difference of biased conductors will remain fixed when conductor potentials change.)

Cond.	BC Type	Value	Biased From
1	Fixed Potential	0.000E+00	0
2	Fixed Potential	0.000E+00	0
3	Fixed Potential	0.000E+00	0
4	Fixed Potential	0.000E+00	0

Figure 2.4. Conductor potential screen.

(4) After achieving the correct array of insulator potentials, signify completion with the <ESC> key. You will be returned to the main IPS pull-down menu.

(5) Choose "Write" to enter your potential conditions into the DynaPAC DataBase files.

You may now wish to change some insulator boundary conditions.

(6) Select "Insulators" from the pull-down menu.

(7) You are given a screen to specify the range of conductor numbers and the range of material numbers to which your changes will apply. (Only insulating materials will be affected.) Alter the screen to suit your wishes, and signify completion with <ESC>.

(8) Select "Fixed" to apply fixed potentials, or "Float" to apply fixed electric field. (Non-zero fixed electric field may be specified.) The remaining choices are "Normal" (to further narrow the range of cells to which the condition is applied) and "Edit" (to specify condition cell by cell for all cells in the range).

(9) Enter the value of potential or electric field. Signify completion with <ESC>.

(10) To see the result of your entries, select "Show Table." The table displays the matrix of conductor numbers and material names with the IPS specification for each matrix cell. The entries are:

Table Entry	Meaning
(Blank)	No such surfaces
(Real Number)	Fixed potential value
FLOA	Fixed electric field (of some value)
MIX	Surfaces have differing specifications.

(11) Choose "Write" to enter your potential conditions into the DynaPAC DataBase files.

IPS writes a file, "ips.out," with a complete cell by cell specification of the initial potentials.

2.4.2. Using DynaPre [Potent] to Create Potential Solver Input

Select "Potent" From the top menu of the DynaPre module. Select "Edit Script" from the resulting pull-down menu. The potential solver input screen shown in figure 2.5 will appear. Modify the input screen to suit your needs. Exit from this screen with an <ESC>. Then select "Make Script" to write a fully commented input deck for the potential solver. Subsequent modifications to the input deck are conveniently done with the text editor.

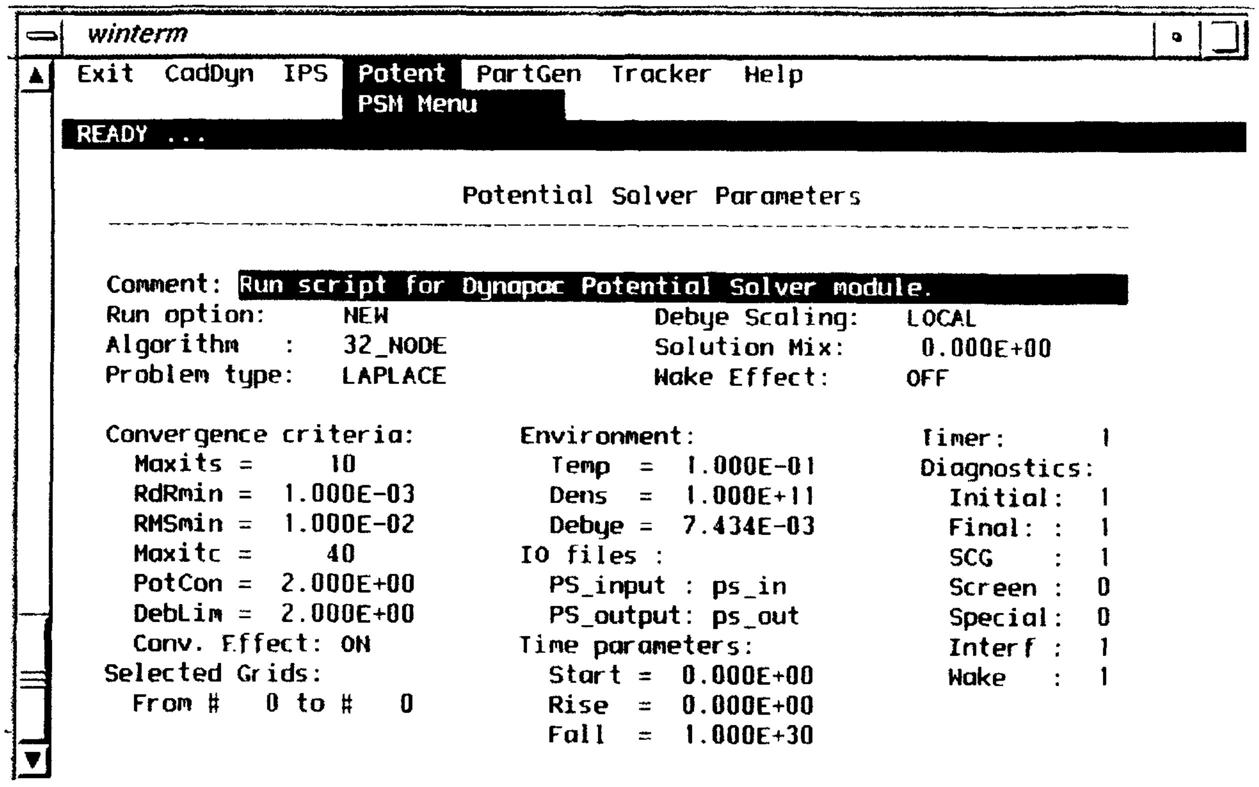
Here is a discussion of the potential solver input parameters.

Run option:

This toggle field takes the values "New" or "Continue." "New" must be specified for the first potential run on a DynaPAC file set. "Continue" can be specified (and is usually preferable) on any subsequent run, even if IPS has been used to change surface boundary conditions.

Algorithm:

The potential solver can operate in "32_Node" (continuous electric field) mode or "8_Node" (trilinear potential) mode. Other modules do not support the "8_Node" potentials, although the potential solution is much faster. It has not been demonstrated that solving potentials with the 8_Node algorithm and then switching to the 32_Node algorithm saves any time.



```
                    Potential Solver Parameters
-----------------------------------------------------------------------

Comment: Run script for Dynapac Potential Solver module.
Run option:     NEW                     Debye Scaling:   LOCAL
Algorithm  :    32_NODE                 Solution Mix:     0.000E+00
Problem type:   LAPLACE                 Wake Effect:     OFF

Convergence criteria:        Environment:                Timer:       1
  Maxits =      10             Temp   =  1.000E-01       Diagnostics:
  RdRmin =  1.000E-03          Dens   =  1.000E+11         Initial:  1
  RMSmin =  1.000E-02          Debye  =  7.434E-03         Final: :  1
  Maxitc =      40           IO files :                    SCG    :  1
  PotCon =  2.000E+00          PS_input : ps_in            Screen :  0
  DebLim =  2.000E+00          PS_output: ps_out           Special:  0
  Conv. Effect: ON           Time parameters:              Interf :  1
Selected Grids:                Start  =  0.000E+00         Wake   :  1
  From #    0 to #    0        Rise   =  0.000E+00
                               Fall   =  1.000E+30
```

Figure 2.5. Potential solver input screen from the DynaPre preprocessor.

Problem Type:

This toggle field describes the space charge function to be used in the potential solver. These are implemented in the subroutine rhoofp.f. Available choices are:

LAPLACE - Zero space charge.

LINEAR - Linear (Debye) screening.

NON_LINEAR - Standard equilibrium space charge formula.

FROZEN_ION - Ion density is everywhere equal to the ambient plasma density. Electron density is $\exp(\phi/\theta)$ for negative potentials.

TRACK_ION - Ion density is taken from particle tracking results (array RHO_Ion). Electron density is $\exp(\phi/\theta)$ for negative potentials.

EXPLICIT - Both electron and ion densities are taken from particle tracking results (array RHO_Ion). ("Full PIC.")

SHEATH_WAKE - Ion and electron densities are calculated with orbital motion (i.e., the wake of the spacecraft sheath and its effect on currents) taken into account.

SPECIAL - Developmental space charge option. Presently, this option is similar to the SHEATH WAKE option, but uses tracked electron densities (array RHO_Elec) in positive potential regions.

Convergence criteria:

Normally only the two parameters "Maxits" and "RMSmin" are varied from their default values.

Maxits - The maximum number of major, or "space charge" potential iterations to be performed.

RdRmin - The value of the "RDOTR" parameter below which the potential will be considered converged.

RMSmin - The value of the "RMSERR" parameter below which the potential will be considered converged.

Maxitc - The maximum number of minor iterations within each conjugate gradient solution.

PotCon - The number of orders of magnitude that the RDOTR drops within each major iteration before it is considered converged.

DebLim - The number of Debye screening lengths allowed per zone. The various space charge treatments limit the amount of space charge in a zone in accordance with this parameter.

Conv. Effect - Whether the analytic trajectory convergence treatment is used in the NON LINEAR problem type.

Selected Grids.

It is possible to apply the potential solver to a subset of the DynaPAC grids. This option has not been fully tested.

Debye Scaling

The "DebLim" parameter (see above) may be applied based on the local grid spacing or on the primary grid spacing.

Solution Mix

For problems which do not converge well this parameter allows admixture of the solution from the previous major ("space charge") iteration at the end of the current major iteration.

Wake Effect

This option is to be turned on to take account of spacecraft motion. It brings down a sub-menu to allow the spacecraft velocity to be specified. On a "NEW" potential run, the wake of the (uncharged) spacecraft surfaces will be calculated. ("Neutral Approximation.") A separate module is used to calculate the wake of the spacecraft sheath.

Environment

A single maxwellian environment is specified in terms of plasma temperature [eV], plasma density [m^{-3}] and Debye length [m]. Only two of these parameters are independent. Debye length will be automatically recomputed following change in temperature or density. Density will be automatically recomputed following change in Debye length.

IO Files

The PS_input name is the name of the file to be written by the "Make Script" selection from the Potent pull-down menu. PS_output presently has no effect.

Time Parameters

For time-dependent problems conductor potentials can be modified (relative to IPS settings) by a pulse-shape factor. This is done in subroutine setcnd.F. The expression currently used is

```
P(t) = PCond*[1.-exp(-(t-t1)/t2)]*exp(-(t-t1)/t3)
where                t1 is start time [seconds]
                     t2 is rise time [seconds]
                     t3 is fall time [seconds]
```

(A rise-time of zero indicates that no time-dependent factor is to be applied.) The time parameter, t, is advanced by the Tracker module.

Timer and Diagnostics

These parameters govern optional printout from various portions of the potential solver.

2.4.3. Operation of the Potential Solver

In this section we discuss the operation of the potential solver from the point of view of monitoring its progress.

It is not recommended that the potential solver be run from within DynaPre. If "Dynapac.setup" has been sourced, run the potential solver with

```
Potent < ps_in > ps_out
```

Otherwise, use

```
$DYNAPAC/Potent_cputype < ps_in > ps_out
```

where cputype is either SUN4 or IRIX.

A potential solver run consists of an initialization phase, a number of major ("space charge") iterations of the potential solution, and a brief exit phase. In the initialization phase, the input parameters are read and echoed, and the DataBase information is processed. Grid information (mesh size and sheath potential) is printed. The grid interface pairs list is formed and printed. Conductor potentials are printed.

At the beginning of each major potential iteration the space charge function and its derivative are evaluated cell by cell, and the conjugate gradient process is initialized. The "initial rdotr" is a measure of the current error in the potential solution. For most cases the "initial rdotr" should decrease monotonically beyond the first few major iterations.

During the conjugate gradient process (governed by subroutine psscg.F) the "rdotr" parameter is printed for each minor iteration. This parameter should generally decrease, but for most cases will not decrease monotonically. The command "grep rdotr ps_out" is a good way to check on the progress of the potential solver. The conjugate gradient process is deemed converged when the criteria discussed above (usually two orders of magnitude decrease in "rdotr") are satisfied.

We are now ready to conclude the major iteration. The most time-consuming task is to calculate and update surface electric fields, which are held fixed during the conjugate gradient process. This requires a full matrix operation (performed by subroutine ecoprd.F), as the electric field value is related to the residual for the corresponding potential. The field updates (subject to several restrictions) are performed by subroutine eupdat.F. The new surface potentials and fields are printed. The difference between the new and previous potential solutions are expressed a root-mean-square errors, and are printed grid by grid ("grep RMS ps_out") and overall ("grep rmserr ps_out"). If the root-mean-square errors remain constant, solution-mixing may ameliorate or solve the problem. (A particularly large "RMS Error" for a particular grid may or may not indicate a problem in obtaining a solution.)

The Potential Solver is ready to conclude when the requested number of major iterations have been performed, or when the "rmserr" has been reduced below its minimum value. By default, the central Z-slice potentials and electric fields are printed for each grid. In these printouts three values are printed for each grid point: the potential value appears at the print position corresponding to the grid point; the x-direction potential gradient [volts per meter] is printed to the right of the grid point, and the y-direction potential gradient above the grid point. The z-direction potential gradient does not appear.

2.4.4. Using Scanner to Display Potentials

The Scanner module is a general interactive utility to extract and display surface-cell or grid structured information from the DataBase. Its main use, however, is to plot potentials.

The top menu bar of Scanner contains the selections "Exit," "Print," and "Plot." With the "Print" selection, virtually any surface or spatial array known to the DataBase Manager can be printed. The print facility is primarily used for diagnostic purposes, and is fairly self-explanatory.

When "Plot" is selected, a pull-down menu is selected with the options "Exit,", "Data Info,", "Edit Plot," "Make Plot", "Show Plot." Each of these options (excluding "Exit") should be selected at least once prior to selecting any of those below. The options chosen for "Data Info" and "Edit Plot" are saved (upon exiting Scanner) in the file "prefix.scan." If you are making a few different plots, you may wish to make a copy of this file for each plot, and copy it back to "prefix.scan" to efficiently regenerate your plot options.

The "Data Info" selection allows you to choose the quantity to be plotted and the normal direction to the plot plane. The "Data Name" field is a toggle field, currently having the options POT_Grid (potentials), RHO_GI (ion densities from wake shadowing calculation), RHO_Elec (tracked electron densities), RHO_Ion (tracked ion densities), NONE (plot gridding only), and OTHER (allowing user specification of another name).

The "Edit Plot" selection gives the user a great deal of control over the plot. Most of the options are reasonably self-explanatory. Be sure that the "Window Manager" and "Plotter" toggle fields are correctly set. Selections under the "Options" title each bring up a sub-menu: "GRID LIMITS" allows the user to specify the region of space to be plotted and the units for the axis labels. "CONTOUR LEVELS" allows specification of up to 24 contour levels (or color boundaries). "CONTOUR MARKS" provides for labeling of contour lines. "LABELS & INCLUDES" controls the labeling and appearance of the plot. "OTHER OPTIONS" at present contains obsolete or little-used choices.

The "Make Plot" selection generates the plot and writes it on the disk file "fort.2." If there are unviewed plots, the new plot will be appended to the "fort.2" file. If there are no unviewed plots, any existing "fort.2" will be overwritten.

The "Show Plot" selection brings up a shell to run the chosen plot display interface. The file "fort.2" will remain after exit from the Scanner module, and can be displayed or printed using any plot interface program in stand alone mode.

2.5. Generating and Tracking Particles

Macroparticles may be used in DynaPAC for a number of purposes including

(1) Studying and/or displaying representative particle trajectories;

(2) Calculating surface currents arising from sheath currents;

(3) Studying wake structure;

(4) Calculating steady-state, self-consistent charge densities;

(5) Calculating time-dependent charge densities and surface currents.

To maximize flexibility, the particle generator and the particle tracker have been developed as two separate modules. The particle generator defines particle species and generates particles throughout volume, along a sheath surface, along a con [illegible] line, in accordance with external input, or along magnetic field lines. The particle tracker will compute the motion of all or a subset of the particles for a maximum time or grid cycling, recording surface currents, and (optionally) plotting trajectories, accumulating steady-state charge density, or calculating new charge density at the updated time. After the particle tracker is run, the particle files are left with updated particle positions and velocities, and the time and cycle number is updated in the DataBase.

Note that there are obvious relationships between the Potential Solver module's "Problem Type" space charge options, the Particle Generator module's "Particle Type" options and the Particle Tracker module's "Process" options. It is pre[illegible]ure to lay down firm rules about which options correspond, but the user should pay attention to the physical processes modeled and the data arrays generated or required in planning a calculation strategy.

2.5.1. Using DynaPre [Par[illegible]en] to Create Particle Generator Input

The particle generator input screen is obtained with the "PartGen/Edit Script" selection from DynaPre. It is used to obtain a fully commented input deck for the PartGen module. The text editor may be used subsequently to modify the input deck.

The main scre[illegible] has selections for the magnetic field, the number of species, and the name, charge, and mass of each species. Density and temperature will be the values used on the most recent potential solver run.

Further description of the particle generation process depends on the "Particle Type" toggle field. The choices are:

DEFAULT

Generate particles (8 per zone) representing a uniform distribution of charge.

SHEATH

Generate particles representing sheath currents. A subscreen will appear to enter the sheath potential.

CONTOUR

Generate particles along a contour line. (This is usually done in order to display particle trajectories.) A subscreen will appear to enter the contour potential and the normal and offset of the cut plane (in primary grid units).

EXTERNAL

Read initial particle data from an external file. A subscreen appears for the file name. The external file is read by subroutine redext.F. The file format is

```
POSITION     x        y     z
DIRECTION    dx       dy    dz
ENERGY       energy
```

where the position is in primary grid units, the direction need not be normalized, and the energy is kinetic energy in electron volts. Alternatively, the keyword E_TOTAL may be used to give the total particle energy. Each such three card sequence results in definition of a particle. A unit particle weight is assigned, on the assumption that this input is used only for trajectory plotting purposes.

B_FIELD

The B_FIELD option is used to generate particles where magnetic field lines enter the computational space. The subscreen is used to enter the particle energy, the (square root of the) number of particles for each boundary surface, and the magnetic field.

2.5.2. Using DynaPre [Tracker] to Create Particle Tracker Input

The particle tracker input screen is obtained with the "Tracker/Edit Script" selection from DynaPre. It is used to obtain a fully commented input deck for the Tracker module. The text editor may be used subsequently to modify the input deck.

The main screen has selections for the magnetic field, the number and selection of species to be tracked, the maximum trajectory time, other limits concerning tracking fidelity, and limits associated with particle origins and trajectory display. There is a slot reserved for the particle pushing algorithm, though at present only one algorithm is implemented (in subroutine movpar.F).

The use to which particle trajectory information is put (usually in subroutine savinf.F) is governed by the "Process" parameter (which is a toggle field). Current choices for the "Process" parameter are:

SPACE_CHARGE

This is the "PIC" option. After each particle has been tracked for the requested time, its charge is shared to the nodes of its current grid cell. The charge is stored in the RHO Ion array. Special cells and surface elements are treated correctly.

TRAJECTORIES

This is the option to track particles for the purpose of either displaying their trajectories or calculating surface currents. A subscreen comes up to select the direction normal to the plot. (Only one plot is produced.) To obtain a plot, the "Plot Limit" option (see below) must be specified; otherwise plotting will be turned off.

PLOT_PARTICLES

With this option no tracking is performed, but the current positions of the particle are plotted. A subscreen comes up to select the direction normal to the plot. (Only one plot is produced.)

TRJ_CHARGE

This is the option to accumulate space charge over the entire trajectory length. The space charge is not shared to nodes, but is accumulated as an element centered array, RHO_Elec. Charge accumulates in empty cells only. Special cells and surface elements are not treated correctly.

Other particle tracker options include:

Tracking Time - The maximum time [seconds] a particle is to be tracked. For time dependent problems, the time variable will be incremented by this amount.

Max Dx - The maximum distance (in local grid units) a particle moves during a substep. Substep distances may be far less than this value. Electrons gyrating in a magnetic field move only for a fraction of the cyclotron period. Slow ions may move less than "Max Dx" in the tracking time.

Max Steps - The maximum number of substeps per particle per iteration.

Max iterations - The maximum number of cycles through the list of grids.

Saving Interval - How often potentials are saved (by subroutine spaini.F) in time-dependent problems.

Particle Limit - Limits (in primary grid units) of initial particle locations. Particles originating outside these limits are ignored. If not set (all zeroes) all particles will be tracked.

Plot Limit - Limits within which particles or trajectories are plotted. If not set (all zeroes) no plot will be produced by the TRAJECTORY process.

Number of Species - Number of species to be tracked.

Species Code - Which species to be tracked (corresponding to species definitions in particle generator input).

B field - Magnetic field.

Mixing Factor - For the TRJ_CHARGE process, the fraction of the previous RHO_Elec values to mix with the newly computed values.

TR_input - Name of file produced by "Make Script" selection.

TR_output - (No effect)

Tracker Diag - Diagnostic level for Tracker subroutines.

Dynalib Diag - Diagnostic level for Dynalib subroutines.

Timer Level - Frequency of cpu time monitoring.

2.5.3. Operation of the Particle Generator

In this section we briefly describe how the particle generator operates for each "particle type," pointing to some of the key subroutines in the process. In general, subroutine inivel.F is used to assign initial velocities to particles. Subroutine convrt.F converts volume weights to charge or current weights.

For the "Default" particle type (IPType=1, uniform density throughout space), particle generation is under the control of subroutine genpar.F. Actual particle generation is done by subroutine fndpar.F. Particle weights are proportional to the shadowing factors, RHO_GI. Particle weights are charge divided by the permittivity of free space, or [volts m^{-2}].

For the "SHEATH" particle type (IPType=2, particles generated on potential contour surfaces) particle generation is under the control of subroutine genpa2.F. Actual particle generation is done by subroutine sthpar.F. SthPar varies its procedure if magnetic effects dominate, as determined by comparing the Larmor radius to the outer grid spacing. Subroutine dozone.F determines the particle positions and area weights, also eliminating high field or bipolar zones. For non-magnetic cases, subroutines inivel.F and dflux.F determine particle velocity and current. For magnetic cases subroutine magsth performs this function, assigning a "cos θ" factor to account for particles traveling along field lines to reach the sheath surface. Particle weights are amperes.

For the "CONTOUR" particle type (IPType=3), particle generation is under the control of subroutine genpa3.F. Actual particle generation is done by subroutine cntpar.F.

For the "EXTERNAL" particle type (IPType=4), particle generation is under the control of subroutine genpa4.F. Actual particle generation (i.e., reading the input file) is done by subroutine redext.F.

For the "B_FIELD" particle type (IPType=5), particle generation is under the control of subroutine genpa5.F. Particles are generated only on the surface of the outermost grid. Actual particle generation is done by subroutine bparts.f. Particle weights are amperes.

2.5.4. Operation of the Particle Tracker

The particle tracker operates on the prefix.PTn files, which are organized in pages of 1000 particles each. Track is kept of how many particles on each page belong to each grid. An "iteration" consists of looping through each grid, and within that through the particles belonging to that grid. Subroutine trkpar.F supervises the tracking of each particle by subroutine movpar.F. Each particle is tracked until one of the following conditions occurs:

(1) The particle strikes the spacecraft;

(2) The particle is found in a grid other than the current grid or its parent, in which case it is transferred to the new grid.

(3) The particle exits the computational space;

(4) The trajectory time reaches the requested particle tracking time;

(5) The number of substeps exceeds the maximum substep number;

(6) For the TRJ_CHARGE process, if the particle is found outside the "sheath," (i.e., in a region where it has very low kinetic energy).

Trajectories stopped for conditions (2) or (5) above are picked up during the next iteration.

After processing each grid Tracker prints the particle number and total weight for each of several particle categories:

(1) "new particles" is the set of particles with which Tracker started;

(2) "partially tracked" refers to particles which have not hit the spacecraft or left the primary grid. These particles may or may not yet have been tracked for the requested particle tracking time.

(3) "dead" particles are those which struck the spacecraft.

(4) "went off primary grid" counts those particles which have exited the computational space.

(5) "trapped" is presently a null category, as there is no criterion for trapped particles. Trapped particles appear as "partially tracked."

(6) "unknown status" is another category of particles with no known means of identification.

When Tracker is finished with the requested number of iterations, it prints the total current to the spacecraft and a table apportioning that current by material and conductor. Also, a "Hit" file is printed listing each particle that struck the spacecraft with its vital statistics which may be used for further processing. The print and format statements for this file may be found in subroutine wripar.F.

If the process involves trajectory plotting, a "fort.2" file will be created containing the plot.

2.6. Using the DynaPost Utilities

DynaPost is a user interface to the Currents and ObjPotl postprocessors.

The Currents postprocessor is intended primarily for time-dependent problems. By reading data from the prefix.CUR file, it can reproduce the table of surface current (sorted by conductor and material number) printed at the conclusion of the Tracker module. It can also create plots of one or more of the table entries as functions of time.

The ObjPotl postprocessor allows the user to display the spacecraft and the results for spacecraft surfaces. Plots produced are

(1) Object outline ("wireframe") plot, color-coded by material

(2) Object with surface potential contour lines drawn.

(3) Object with surface cells shaded by material.

(4) Color-coded object surface potentials.

(5) Color-coded surface normal electric fields.

(6) Object with surface cells shaded by conductor number.

(7) Object with surface cells shaded by incident current.

(8) Color-coded incident flux densities.

When ObjPotl is chosen from the top-level DynaPost menu, the user is presented with a screen from which he can vary input options to ObjPotl, create an input file, run (yes, this one actually works!!) ObjPotl, and view the resulting plot file. General options include a title for the plot, a vector (unnormalized) from the spacecraft toward the viewer, and the plot display interface to be used. By default, surface cell numbers appear on plots (1), (3), and (6) above; toggles to "ALL" or "NONE" control the appearance of these numbers. For each plot a "YES/NO" toggle is available to determine whether the plot is made; in addition plots with no information (e.g., uniform potential) will not be made. For the potential, field, current and flux plots, the range of the color table may be restricted; by default the range with be the range of actual values found.

The run controls appear in the lower left of the screen. "MAKE SCRIPT" writes a fully commented input deck as "objpotl.in." "RUN SCRIPT" issues (in either foreground or background) the command "ObjPotl < objpotl.in > objpotl.out," creating a "fort.2" plot file. "SHOW PLOT" spawns a shell to run the selected plot display interface.

On exit a plot file "fort.2" will remain for disposition by any of the plot display interfaces, and the input and output files from ObjPotl will remain. If the incident flux plot has been made, the area, current, and flux to each surface cell will be listed in the file "ObjPotl_Currents."

2.7. Time-Dependent Problems

DynaPAC is intended for time-dependent problems. However, it must be confessed that only one complex time-dependent problem (involving mobile ions and barometric electrons) has ever been run. It is to be expected that any application not entirely similar to that one will require some code modification. In this section we remind the user of the time- dependent features of DynaPAC, most of which were mentioned above. Remember that the bulk of a time-dependent problem consists of alternate Potent-Tracker runs.

2.7.1. Setting Initial Potentials

In DynaPre/IPS, potentials may be set to fixed values, or they may be set to "Bias" values and the "Adjust" options used. In the former case (fixed values), the pulse-shape options of Potent may be used to create a time-dependent multiplier for the fixed potentials. In the latter case ("Adjust" options) the overall object potential may change due to currents incident on the spacecraft.

2.7.2. Running Potent

Potent should be run with appropriate "Problem Type", such as "TRACK_ION" or "EXPLICIT." However, "FROZEN_ION" or "LAPLACE" may be more appropriate to the initializing run. Remember to set pulse shape parameters if desired, and, after the first run, to set the "New or Continue" parameter to "Continue." Also, for the "Continue" runs you probably will want to set "Maxits" to a low value (such as two or three).

2.7.3. Running PartGen

Normally, for time-dependent problems PartGen will be run after the initial Potent with the "DEFAULT" particle type, corresponding to a uniform charge distribution. There is currently no provision for refreshing the particles as they are drained from the computational space. There also are no current provisions for secondary particles at surfaces or for ionization products.

2.7.4. Running Tracker

Tracker is the key module for time dependent problems. With the "SPACE_CHARGE" process, a Tracker run will update the time and timestep number, save the surface currents (in the prefix.CUR file), and (in accordance with Tracker input) save the spatial and surface potentials.

2.7.5. Displaying Results

Scanner can display any of the potential arrays saved by the Tracker module.

DynaPost can form tables and plots of surface currents or their time histories.

Tracker may be used with the "PLOT_PARTICLES" process to display current particle positions. (This is usually done immediately following the Tracker/SPACE_CHARGE run.)

3. DynaPAC Developer's Manual

This section is intended primarily for users who would like to modify or extend the capabilities of DynaPAC. There is not much in this initial version, as the contract has run out of steam.

3.1. DynaPAC Software Structure

DynaPAC has three main portions contained in three directories:

The "dynapac" directory contains all of the physics modules, as well as many of the utilities. This directory corresponds to the DYNAPAC environment variable.

The "ScreenPkg" directory contains the low-level routines for handling the interactive screens.

The "Plotters" directory contains the plot display interface routines.

In this manual we will only be concerned with the "dynapac" directory.

3.1.1. DynaPAC Directory Tree

In this section we describe contents of the $DYNAPAC directory and its sub-directories. We use an appended slash (/) to indicate a directory; absence of a slash indicates a file.

DynaPAC.setup - This is the file establishing environment variables and aliases for a DynaPAC user or developer.

Makefile - This should make all the DynaPAC modules.

bin/ - Directory for DynaPAC executable modules and shell scripts.

dynamake - Shell script which runs DynaPAC Makefiles.

inputs/ - Contains screen definitions and defaults for the DynaPAC interactive modules.

lib/ - Contains DynaPAC libraries.

src/ - Contains the source code, object files, and include files for the DynaPAC modules and libraries.

src/AddGrid/ - Code to add an outer grid to a DynaPAC grid structure.

src/DynaPost/ - Code for the DynaPost module.

src/DynaPre/ - Code for the DynaPre module.

src/GridTool/ - Code for the GridTool module.

src/NisDyn/ - Code for object definition interface to EMRC DISPLAY-II.

src/ObjPotl/ - Code for the ObjPotl module.

src/PartGen/ - Code for the PartGen module.

src/PatDyn/ - Code for the PatDyn module.

src/PolDyn/ - Code for the PolDyn module.

src/Potent/ - Code for the Potent module.

src/Scanner/ - Code for the Scanner module.

src/Tracker/ - Code for the Tracker module.

src/cadlib/ - Code for the library of object definition interface routines.

src/currents/ - Code for the library of routines used in the DynaPost/Currents module.

src/dblib/ - Code for the DataBase Manager library routines.

src/dynalib/ - Code for utility routines used by PartGen and Tracker.

src/ipslib/ - Code for routines used by DynaPre/IPS.

src/s3utils/ - Code for the low-level utility routines used by all DynaPAC modules

src/*/Dbx/ - Source code resulting from processing by the "cpp" preprocessor.

3.1.2. DynaPAC Software Conventions

There are many DynaPAC software conventions. Sometimes they are followed. Other times they are not.

Always use "syopen" to obtain a logical unit number and open a file. Get rid of it with "syclos".

3.1.3. Modifying DynaPAC Subroutines

Always type "../../dynamake" to remake a module.

In any of the $DYNAPAC/src/*modulename* directories you will find source files with suffices .f, .F, and .h.

If you modify a .f subroutine, you may compile it in the usual way, then type "../../dynamake" to remake the module. If you do not manually compile it, it should be compiled by the Makefile.

If you modify a .F subroutine, the Makefile should run the "CPP" processor to produce Dbx/name.f, compile Dbx/name.f, and remake the module.

If you alter a .h include file, the Makefile should reprocess all the necessary routines. Note that occasionally include files are shared between modules.

If you add a routine, be sure to add it to the relevant list in the Makefile.

If you alter a library routine, you may manually compile it and add it to the library, or you may type "../../dynamake" to take the library apart, compile the routine, and reconstruct the library.

3.2. Using the DataBase Manager

The following description of the DataBase Manager appeared in the Interim Report, SSS-DPR-90-11973, 30 September 1990. Check in the DynaPAC modules for further examples on the use of the DataBase Manager.

3.2.1. Data Base Library

The data base library provides the analysis and utilities functions necessary to define, manage, store, and retrieve data. In addition to the three callable routines (dbinfo(), dbdata(), and dbfile()) normally used to integrate the data base into an application, a stand-alone driver (dbtool) is also available to interact directly with the data definitions, data, and their files.

When a DynaPAC module is linked to the data base library, the following routines provide memory, data, and file management. The dbinfo() routine is used to define data base building blocks. Typical information commands define grids, grid nesting structures, data, lists, and elements. The dbdata() routine is used to allocate/release reusable memory, read/write from disk files to memory, and initialize data values in memory. The dbfile() routine provides the means to manage the data files and their contents.

The input to these routines is in the form of strings containing keyword commands. The output is returned to the calling routine via common blocks within specialized include files. The calling routine saves the information it requires in its own data structures.

The data base can be thought of as a utility driven by its own keyword input language. Although some defaults are built into the data base, in most cases, the data required to perform a calculation must be constructed before it can be used. The dbinfo() routine is used to define data and the dbfile() routine is to define files.

The reader is referred to the data base header file, DBhead, for a complete keyword definition and current implementation status. Some examples of the use of the library routines are presented below. The first three demonstrate some data base functions. The final two examples show how a calculation using data in the data base memory storage area might be used.

3.2.2. Example: File Definition

The Task

A calculation wants to open a set of DynaPAC files that have the prefix "Demo_Files" and then close them when it finishes.

The Solution

To open the files, use the command

```
call dbfile("open prefix=Demo_Files")
```

This command assumes the default file type is "DYNAPAC." A more explicit command would be

```
call dbfile("open prefix=Demo_Files file_type=DYNAPAC")
```

To close just the Demo_Files file set, use

```
call dbfile("close prefix=Demo_Files")
```

If desired, errors opening or closing the files can be checked by examining the error flags in the dbfile() output include file, "odbfile.h". The logical flag, "ldferr", is TRUE if an error was detected. The character string, "cdferr", will contain an error message after error detection.

3.2.3. Example: Definition of List Dependent Data

The Task

Define a Surface_List list type data item, "Surface_Centroids".

The Solution

To define a data item that is a list, both the list type and the data item must be defined.

Two definitions are helpful since the list type definition depends on the particular object being defined, while this data item does not. Since the number of surfaces, or length of Surface_List is not given, let's define the data item in steps first.

Define the data type of Surface_Centroids.

```
call dbinfo("Define Data=Surface_Centroids Data_Type=LIST")
```

And it is which kind of LIST?

```
call dbinfo("Define Data=Surface_Centroids List_Type=Surface_List")
```

Each surface in the list has three real values, a location in three space or an offset from one of the corners. It depends on how the data is used by the application program. The data base does not know the difference.

```
call dbinfo("Define Data=Surface_Centroids Data_Dim=3 value_type=REAL")
```

All three of these commands could be combined into a single command. (Pretend all of the keywords are in one string and do not have embedded carriage returns.)

```
call dbinfo("Define Data=Surface_Centroids Data_Type=LIST
    List_Type= Surface_List Data_Dim=3 value_type=REAL")
```

Later when we find out how many surfaces there are on this object, the Surface_List can be defined. If there are 1,001 surfaces, then the data definition can be completed.

```
call dbinfo("Define List_Def=Surface_List List_Length=1001")
```

3.2.4. Example: Definition of Grid Dependent Data

The Task

Define a grid dependent data item, "Ion_density", which is element-centered.

The Solution

As in the example before, the data item definition is independent of the grid and element definitions. The data is defined as being of data type, "SPATIAL". This implies a different set of data is required for each grid in the problem. SPATIAL data is defined on the level of an element. Grids are thought to be rectangular lattices with equal spacing in each direction. An element is located at each lattice point. One of the predefined data items is one called "ELEMENT_CENTERED". We can save some work defining Ion_density by taking advantage of an existing definition using the "SAME_AS keyword".

```
call dbinfo("DEFINE Data=Ion_density Same_As=ELEMENT_CENTERED")
```

The element type, ELEMENT_CENTER, is also predefined.

3.2.5. Sample Routine 1

This routine demonstrates how to use statement addressing functions to use the data in the data base manager.

```
        subroutine foo
        ...
        include "cdbdata.h"
        include "cdbinfo.h"
        include "dbmdata.h"
        ...
cc
cc *define local variables
        character*80 scomnd
        ...
cc
cc *define statement functions (this is probably wrong!)
        iaddrs(ix,iy,iz) = iofset+((iz-1)*ny+(iy-1))*nx+(ix-1)
cc
cc *find out about grids, save the number of grids
        call dbinfo("Inquire Problem")
        ngrids = idingd
cc
cc *loop on grids
        do 700 igrid = 1, ngrids
cc
cc *read in potentials for this grid
           write(scomnd,2000) igrid
 2000      format( 'READ DATA=pot_20_node grid=',i6)
           call dbdata( scomnd )
cc
cc *grab stuff out of output common (cdbdata.h) which will be useful later
           iofset = idboff(1)
cc
cc *get information about the potentials in this grid
           write(scomnd,2500) igrid
 2500      format( 'INQUIRE DATA=pot_20_node grid=',i6)
           call dbinfo( scomnd )
           nxpot = idihir(1)
           nypot = idihir(2)
           nzpot = idihir(3)
cc
cc  *loop on nodes of potentials
           do 600 iz=1,nzpot
              ...
 600       continue
cc
cc *all done with this grid, write it out and release its allocated memory
           write(scomnd,7000) igrid
 7000      format( 'WRITE CLOSE DATA=pot_20_node grid=',i6)
           call dbdata( scomnd )
 700    continue
        ...
        return
        end
```

10.5 SAMPLE ROUTINE 2

This routine demonstrates how to write a wrapper routine to hide the data base completely from the calculation routine, foobar().

```
       subroutine foo
       ...
       include "cdbdata.h"
       include "odbinfo.h"
       include "dbmdata.h"
       ...
cc
cc *define local variables
       character*80 scomnd
       ...
cc
cc *find out about grids, save the number of grids
       call dbdata("Inquire Grid=ALL")
       ngrids = ndbgrd
cc
cc *loop on grids
       do 700 igrid = 1, ngrids
cc
cc *read in potentials for this grid
          write(scomnd,2000) igrid
 2000     format( 'READ DATA=pot_20_node grid=',i6)
          call dbdata( scomnd )
cc
cc *grab stuff out of output common (cdbdata.h) which will be useful later
          iofset = idboff(1)
cc
cc *get information about the potentials in this grid
          write(scomnd,2500) igrid
 2500     format( 'INQUIRE DATA=pot_20_node grid=',i6)
          call dbinfo( scomnd )
          nxpot = idihir(1)
          nypot = idihir(2)
          nzpot = idihir(3)
cc
cc *call the calculation submodule
          call foobar(rdata(iofset),nxpot,nypot,nzpot)
cc
cc *all done with this grid, write it out and release its allocated memory
          write(scomnd,7000) igrid
 7000     format( 'WRITE CLOSE DATA=pot_20_node grid=',i6)
          call dbdata( scomnd )
 700   continue
       ...
       return
       end
       subroutine foobar(pot20s,nx,ny,nz)
       real pot20s( nx, ny, nz )
       ...
       return
       end
```

This file (DBhead) contains the headers for calls to data base routines.
The present data base routines are:

```
dbfile( ccomnd ) :  File manipulation commands.
dbdata( ccomnd ) :  Data handling commands, movement from disk memory.
dbinfo( ccomnd ) :  Access to data about data.  Grid and data type info.
```

In general, data base commands are given in keyword oriented strings. The order of the keywords is not important. The keywords are not case sensitive. The command strings can be any length.

Valid names of files and data types start with a letter followed by any combination of letters, numbers, and the character "_". Case is important for file names, but data names are converted to uppercase. Names are stored in 80 character variables presently, but file names on some machines will be truncated to something shorter. Note that a four character suffix will be added to file names in order to identify the format of their contents.

Data item names may have multiple fields. The name for a piece of data will be the data type name, followed by the rest of the fields separated by one space. This constructed name will be truncated to 80 characters. If truncation of a name occurs, an error message is generated. When defining a data item with multiple names, just use the first name to set its properties. Then when making dbdata() calls or inquires, use the entire name.

For now, spatial items are 3 dimensional. When the keyword reader gets smarter, keywords using input lists will use () to enclose the list. For example, "EDGE_LENGTH=(len_x,len_y,len_z)".

Output is returned via a include file. It is unwise to rely on the order of the contents of the include files since they may change. The names of the variables in the file should be constant.

It should be noted that the current implementation of time names has been done in such a way that it may be a useful guide for future modifications. Time stamps are simply names which can be used to group otherwise identically identified data items. If another or additional distinguishing names were desired, the time stamp/name usage could be generalized to allow a number of different and/or parallel sets of data grouping. The relationships of the groupings would be determined by a set of submodules or functional relationships.

BUGS: All errors are fatal right now.
Not all commands are fully implemented.
Defaults are not always documented.

```
The following notes are subject to change as development continues, but
        should be fairly stable.

The keywords have been marked to indicate implementation status.
        No mark means it is implemented and may even work.
        % - Partially implemented.  Restrictions apply.
        + - Partially implemented.  Recognized, but ignored.
        * - Not implemented.  Will generate fatal error.

Keyword defaults are in [ ]'s, where appropriate.

The predefined items are described at the end of the file.

Some example routines using the data base calls are available in the
        dynapac directory on s3dawn in ~dynapac/Notes/Examples.

---

Files Oriented
        dbfile(ccmnd) - interface routine for file handling
                ccmnd is a keyword oriented input string like:
                   "Open Prefix=Dmsp"
                   "STATUS prefix=Data_set_1 PREFIX=Data_set_2"
                output from the routine is returned in "odbfile.h"

        command keywords are :
                EXIT - Close all open files.
                OPEN - Assign file prefix(s) to run.
                CLOSE - Close file prefix(s).
        *       STATUS - Query current file status of prefix(s).
        +       WHAT - STATUS of all known prefixes.
        +       HELP - This list.
        identifying information:
                PREFIX=File_Name - Define a file prefix. [last prefix used]
                FILE_TYPE=keyword - Specify a file type.  (DYNAPAC)
                SAVE_FILE=logical - save file when done[TRUE] or delete(FALSE)
        +       DIAGNOSTIC=keyword - turn ON,[OFF] diagnostics

---

Data Oriented
        dbdata(ccmnd) - interface routine for data base requests
                ccmnd is a keyword oriented input string like
                   "read data=pot_20_node grid=2"
                output from the routine is returned in "odbdata.h"
        command keywords are broken into groups below:
          action instructions:
                OPEN - Allocate space for data in memory.
                CLOSE - Deallocate space for data in memory.
        +       CRUNCH - Squeeze the gaps out of memory.
        %       READ - Transfer data from disk to memory.
                        (% - must open disk prefix first)
        %       WRITE - Transfer data from memory to disk.
                        (% - must open disk prefix first)
        +       COPY - Copy first data name to rest of data names (in memory).
```

```
          INITIALIZE - Initialize a data type to its "set_value".
                 ( - only dimension=1 items)
+         DELETE - Remove a data type from disk.
                 (* - empty space in file is not reclaimed)
          OVER_WRITE - Replace data on disk defined by OLD_DATA, OLD_GRID,
                 and/or OLD_TIME with the data defined by DATA, GRID,
                 and/or TIME.  Will use names of new data (DATA, GRID,
                 and TIME) as defaults for undefined old data
                 variables (OLD_DATA, OLD_GRID, and OLD_TIME).
+         DIAGNOSTIC=keyword - turn ON,[OFF] diagnostics
*         HELP - This list.
   identifying information:
          DATA=name - Specify a data name (repeatable).
          GRID=name - Specify a grid number or "ALL" grids.
          TIME=name - Specify the time stamp.
          PREFIX=file_name - Specify a file prefix.
          OLD_DATA=name - Specify old data name. (See OVER_WRITE
          OLD_GRID=name - Specify old grid number. (See OVER_WRITE
          OLD_TIME=name - Specify the old time stamp. (See OVER_WRITE.
   optional commands to override data base default actions:
          LENGTH=int - Size in words needed for data in memory.  Used
                 to modify data type definition temporarily.  Normally
                 this keyword is only used to OPEN data items with
                 multiple field names.  In this case, the data type
                 is defined with by the first field only while locations
                 in memory are associated with the full name.
          HERE=int - Point to a memory address, use idbloc() to get
                 offset from idata(0).  Used to tell the data base where
                 something is.  Note that much of the internal error
                 checking will be defeated by this command.
```

```
Grid and Data Definition/Type Oriented
        dbinfo(ccomnd) - interface routine for defining and learning
                information about grid definitions and data typing
                properties.  Sorry, only one PROBLEM, GRID, ELEMENT, DATA,
                or INTERP per call.  Use dbdata("STATUS DATA=stuff GRID=2")
                to find the memory offset for the "stuff" data in grid 2.

                a dbinfo() ccomnd is a keyword oriented input string like:

                        "DEFINE Data=pot_8_node element_type=Node_8
                             dimension=scalar value=real set_value=0. "

                output from the routine is returned in "odbinfo.h"

        command keywords are :
                DEFINE - Define or redefine a piece(s) of information.
                INQUIRE - Request information about a GRID, ELEMENT, DATA, or
                          LIST_DEF, INTERP.  (see context keywords)
+               DIAGNOSTIC=keyword - turn ON,[OFF] diagnostics
*               HELP - This list.

        context keywords:
                PROBLEM - Get the generally useful stuff about problem.
        DATA=name - Commands are in regards to data type "name".
                if Command = DEFINE then only first name field is needed
                if Command = INQUIRE then use all name fields to get
                        specific information.  Otherwise, just get
                        general information.
        ELEMENT=name - Commands are in regards to element type "name".
        LIST_DEF=name - Commands are in regards to list type "name".
        GRID=name - Commands are in regards grid type "name".
                Only one grid's worth of information.
*       INTERP=name - not implemented. "linear","quadratic".
        SAME_AS=name - The new thing is the "same as" name.

DATA parameters:
        DATA_TYPE=keyword - this data is SPATIAL, BUFFER, LIST, etc
        DATA_LENGTH=integer - number of elements in a BUFFER
        ELEMENT_TYPE=element_name - flag describing what this value is
        LIST_TYPE=list_name - name of list definition for this data type
        DATA_DIM=dim_flag - either a number (values/pt.) or flag
        VALUE_TYPE=value_type - type of values at each point
                value_type = <REAL/INTEGER/LOGICAL/OCTAL/ALPHA>
        ALPHA_LENGTH=nchars - length of ALPHA types in characters [80]
%       SET_VALUE=value - default initialization value/method.
                (% - only know how to set DATA_DIM=1 to constants)
        SAVE_DATA=logical - save data on perm.[TRUE] or temp.(FALSE)
        OWN_FILE=keyword - [FALSE] - put this data in the main file
                         - TRUE - same as DYNAPAC below
                         - DYNAPAC - put data in own standard subfile
                         - LONG_NUMBER <number_subkeys> - for data
                           types with many subkeys ( >~200).
                           <number_subkeys> is maximum number of
                           keys to put in the file (default is
                           1000).  Addressable only with
                           DATA_NAME <number>, where <number> is
                           between 1 and <number_subkeys>.
        FILE_SUFFIX=name - name to use for OWN_FILE suffix (should
                         include "." as first character)
        GRID_DEPEND=logical - TRUE if data is grid dependent. [FALSE]
        TIME_DEPEND=logical - TRUE if data is time dependent. [FALSE]
-       TIME_SAVE=integer - how many time steps of data to save [1]
                Automatically OVER_WRITEs oldest (smallest time stamp)
                version of data.

ELEMENT parameters:
        UNIT_SIZE=integer - how many sets of values are in each unit
        UNIT_OFFSET=offset_x,offset_y,offset_z - offset in mesh units
                        from element origin, should be of approp.
                        dimension and there should be one offset for
                        each "UNIT_SIZE" for SPATIAL items.
-       INTERP_TYPE=name - method of interpolation between points

LIST_DEF parameters:
        LIST_LENGTH=integer - number of entries in the list_name

INTERP parameters:  none for now, just the name.  Will recognize
        "linear", "quadratic".
        maybe later will add ax**3+bx**2... type stuff.
```

```
GRID parameters (note GRID is a list of SPATIAL elements):
                OUTSIDE - this grid encloses the other grids(if any)
                INSIDE=grid_name - this grid is inside grid_name
                GRID_SIZE=real - grid mesh size (meters), redefines all grids!
                MESH_RATIO=integer - ratio of inner/outer mesh units (>=1)
                PRIM_RATIO=integer - ratio of inner/primary (or outer most)
                               mesh units
                ORIGIN=value_x,value_y,value_z - In the INSIDE grid the
                               origin is in outergrid units.  For OUTSIDE
                               grids, the origin is the location of the old
                               grid 1 in this grid.
                               ( always true: ORIGIN=1,1,1 in outermost grid )
                ORIGIN_PRIM=value_x,value_y,value_z - Origin location in
                               primary or outer most grid units.
                EDGE_LENGTH=length_x,length_y,length_z - number of nodes
                               along the grid edge, includes endpoints.
%               GRID_INTERP=name - outer grid interpolation function
                       (% - =0 for now)

---

The following items are predefined :
( <none> means no default value is set for that attribute )

Data Types [OwnFile=FALSE]
   BUFFER Data types
        Name            DatTyp  ValType  Len SaveDat InitVal Dim GrdDep TimDep
        BUFFER_DATA     BUFFER  INTEGER  0   FALSE   ' 0'     1  FALSE  FALSE

   LIST Data types
        Name            DatTyp  ListTyp ValType SaveDat  InitVal Dim GrdDep TimD
        LIST_DATA       LIST    <none>  INTEGER TRUE     ' 0'     1  FALSE  FALSE

   SPATIAL Data types [ValType=REAL, InitVal=' 0.0', SaveDat=TRUE, DataDim=1,
                       GridDep=TRUE, TimeDep=FALSE, NumTimSav=1]
        Data Type Name          Element Type Name
        SPATIAL_DATA            <none>
        NODE_8                  NODE_8_ELEMENT
        NODE_20                 NODE_20_ELEMENT
        NODE_32                 NODE_32_ELEMENT
        ELEMENT_CENTERED        ELEMENT_CENTER

Element Types
        Name            Pts/Elm         Interp          Pt Locs
        ELEMENT_CENTER  1               LINEAR          (0.5, 0.5, 0.5)
        NODE_8_ELEMENT  1               LINEAR          (0.0, 0.0, 0.0)
        NODE_20_ELEMENT 4               QUADRATIC       (0.0, 0.0, 0.0)
                                                        (0.5, 0.0, 0.0)
                                                        (0.0, 0.5, 0.0)
                                                        (0.0, 0.0, 0.5)
        NODE_32_ELEMENT 4               QUADRATIC       (0.0, 0.0, 0.0)
                                                        (0.5, 0.0, 0.0)
                                                        (0.0, 0.5, 0.0)
                                                        (0.0, 0.0, 0.5)
List Types (Lengths should be redefined)
```

```
        Name              Length
        SURFACE_LIST      0
        SURFACE_NODES     0

Grids (the outermost grid should be redefined)
        Name     MeshLen Origin            EdgeLen          Grid_Interp
        '1'      1. m    (1., 1., 1.)      (7, 9, 11)       NONE

---
end of file "DBhead"
```

3.3. Modifying Screens

The coding for the screens exists as ascii files in $DYNAPAC/inputs/*/*.sc. It is possible to modify these. There is currently no documentation on this process.

3.4. Potential Interpolation Scheme

3.4.1. Continuous Field Interpolants

We originally proposed using triquadratic interpolants in DynaPAC in order to obtain more nearly continuous electric fields than were provided by the trilinear interpolants used in previous three dimensional codes. Experience has shown that the proposed method suffers from the two disadvantages that (1) the fields are still not strictly continuous; and (2) the difference in weight between the corner nodes and the edge-center nodes leads to a poorly conditioned matrix. Therefore, we have abandoned the triquadratic formulation in favor of a finite element interpolation scheme which has strictly continuous electric fields.

The basic interpolation functions consist of functions which have unit value and zero slope at their home nodes, and zero value and slope at opposite nodes:

$$F_0 = (z-1)^2/(1-2z+2z^2)$$
$$F_1 = (z)^2/(1-2z+2z^2)$$

and functions which have zero value and unit slope at their home nodes and zero value and slope and opposite nodes:

$$G_0 = z(1-z)F_0$$
$$G_1 = z(z-1)F_1$$

The functions F_0 and G_0 are shown in figure 2.6. It can be verified that these functions can exactly reproduce constant, linear, and quadratic potentials.

We generalize to three dimensions by assigning to each node of the unit cube four interpolation functions corresponding to the potential, x-component of potential gradient, y-component of potential gradient, and z-component of potential gradient for that node. Thus, the contribution of these four quantities at point [0,0,0] to the potential at [x,y,z] is governed by the interpolation functions $F_0(x)F_0(y)F_0(z)$, $G_0(x)F_0(y)F_0(z)$, $F_0(x)G_0(y)F_0(z)$, and $F_0(x)F_0(y)G_0(z)$. (Contributions of other cube corners are obtained by changing "0" subscripts to "1" as appropriate.) We elect to omit terms of the form GFG or GGG.

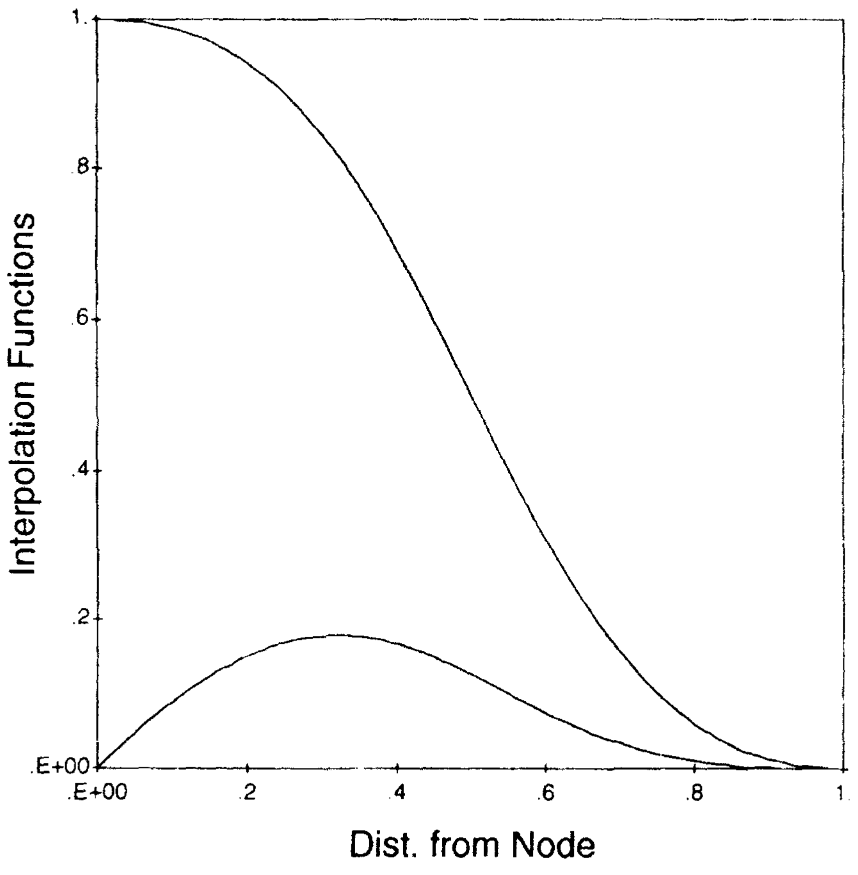


Figure 2.6. Interpolation functions F_0 and G_0.

3.4.2. Interpolating Potentials and Fields for Special Elements

Special elements are those elements which contain surfaces. Because surface-containing elements are not empty cubes, the potential interpolation functions described in the previous section cannot be straightforwardly applied. To develop finite element matrices for these cells, we require a prescription for calculating the potential and electric field in the cell interior. Note that a "special element" is bounded by three types of surfaces: (1) square surfaces bounded by grid edges and shared with adjacent (presumably empty) elements; (2) object surfaces; and (3) surfaces bounded by both object points or edges and grid points or edges. On type (1) surfaces we must use the potential and field interpolation described in the previous section. On object surfaces, the potential and electric field must be expressed in terms of the object's surface cell potentials and normal fields. Type (3) surfaces must smoothly blend between the two. We describe below an algorithm to express the potential and field at any point in the cell volume in terms of the grid point potentials and fields, and the surface cell potentials and normal fields. The algorithm has the property that, when applied to a cell with six type (1) surfaces, constant, linear, and quadratic potentials can be represented exactly.

Let **R** be a point in a volume bounded by a set of surfaces $\{S\}$, each of which may be a triangle or a planar convex quadrilateral. For a given S, let **P** be the point on S nearest **R**, let $\phi(\mathbf{P})$ and $\mathbf{E}(\mathbf{P})$ be the potential and electric field at **P**, and **n** be the unit normal (from the surface point **P** into the volume) to the surface. Let $A(S)$ be the area of the surface. Let

$$\begin{aligned} &\mathbf{d} = \mathbf{R} - \mathbf{P} \\ &d^2 = \mathbf{d} \bullet \mathbf{d} \\ &K(S) = A(S)/\, d^2 \text{ for } \mathbf{d} \bullet \mathbf{n} > 0 \\ &N = \Sigma_{\{S\}}\, K(S) \end{aligned}$$

Let l be the distance from **P** in direction **n** to the next surface intersection. (In practice, it is adequate to extend l to the intersection with the surface of the rectangular parallelepiped bounding the volume.) Then the contribution of S to the potential at **R** is

$$\phi_S(\mathbf{R}) = (K(S)/N)\, [\phi(\mathbf{P}) - (1 - \mathbf{d} \bullet \mathbf{n}/l)\, (\mathbf{d} \bullet \mathbf{n})\, (\mathbf{E}(\mathbf{P}) \bullet \mathbf{n})]$$

and the total potential is found by summing the contributions from all the surfaces.

The electric field is found by differentiating the above. The contribution of S to the electric field is

$$\begin{aligned} \mathbf{E}_S = {} & \phi_S\, [\nabla N/N - \nabla K(S)/K(S)] \\ & + (K(S)/N)\, [-\nabla\phi(\mathbf{P}) + (1 - 2\, \mathbf{d} \bullet \mathbf{n}/l)\, \mathbf{n}\, (\mathbf{E}(\mathbf{P}) \bullet \mathbf{n})] \end{aligned}$$

where $\nabla\phi(\mathbf{P})$ is taken tangential to the surface, so that

$\nabla\phi(\mathbf{P}) = 0$ for $\mathbf{P}$ at a vertex of S

$\nabla\phi(\mathbf{P}) = -\mathbf{e}\,(\mathbf{e} \bullet \mathbf{E}(\mathbf{P}))$ for $\mathbf{P}$ on edge with direction $\mathbf{e}$

$\nabla\phi(\mathbf{P}) = -\mathbf{E}(\mathbf{P}) + \mathbf{n}\,(\mathbf{E}(\mathbf{P}) \bullet \mathbf{n})$ for $\mathbf{P}$ in surface interior

To implement this we interpolate surface fields and potentials on bounding surfaces in a manner which maintains the continuous field property. Potentials and normal field components are defined at the centroids of the original object surfaces and area-weighted averaged at surface corners. Potentials, normal fields, and tangential field components at all surface points which are vertices of bounding surfaces are expressed as linear combinations of centroid potentials and fields. This transformation is given by the matrix TNdCnt in the special element's "Bound_Surfs" record.
The format of the "Bound_Surfs" record is:

Word	Variable	Contents
1	NCSurf	Number of bounding polygons
2	NCNode	Number of nodes(a)
3	NCnts	Number of centroids
4	LTrans	Length of constraint relation matrix
next 8*NCSurf	JCSurf(8,*)	Bounding polygon node list
next 3*NCNode	RCNode(3,*)	Node locations
next NCnts	MatCnt(*)	Centroid list
next LTrans	LTCnt(*)	Constraint relation map
next LTrans	TNdCnt(*)	Constraint relation matrix

(a) Nodes 1-8 are cube corner nodes. Remaining nodes are surface cell centroids, surface cell corners (constrained), and additional surface nodes (constrained).

3.5. Surface Potential Boundary Conditions

Physically, it is possible to specify, for each surface element, either its potential or its surface normal electric field. (For conductors, the choice is between potential and total charge.) From the point of view of the interpolation functions (and therefore the potential equations), however, the potential and surface field appear as independent variables. DynaPAC legislates the physical constraint that one or the other must be specified.

Experience has shown that the surface electric fields are weakly coupled to the potential equations, so that solving for the surface electric field variable does not give reliable results. Rather, we always solve Poisson's equation with all surface electric fields fixed. To determine surface field values for fixed potential cells, we use the fact that the surface cell charge is conjugate to its potential, i.e., when the full conjugate gradient matrix multiplication is performed, the surface cell charge (corresponding to the existing potentials and fields) appears in the surface-cell-potential slot of the "AU" vector. This is an improvement over "capacitance matrix" methods in that it takes full account of the nonlinear space charge in the external space.

The matrix operations referred to above are carried out by Subroutine ECoprd. The surface cell charge values are then passed to Subroutine EUpdat to determine new surface electric field values. The new value is first determined by dividing the charge by the cell area. Then, EUpdat mixes 30% of the previous (old) values with the newly calculated values. Finally, EUpdat bounds the allowable electric fields based on the cell dimensions, the minimum mesh size in the problem, and the range of potentials in the problem. A diagnostic is printed for those cells whose calculated electric fields fall outside the allowable range, and the unper or lower bound (as appropriate) is used for the new electric field value.

Most of this machinery appears to work. DynaPRE/IPS may be used to set potential values on conductors, and potential or electric field values on insulating surfaces. DynaPRE/IPS may also be used to define a floating, biased set of conductors numbered consecutively starting with "1", with fixed potentials on higher numbered conductors. Nonzero charge values for these conductors may be generated using Tracker, or entered manually using a stand alone module.

3.6. Space Charge Treatments

The various space charge treatments available in the DynaPAC potential solver were enumerated in Section 2.4.2. Here we give some addition details for each of the options. These treatments are implemented in subroutines RhoOfP and PSSern. RhoOfP returns values to be used for the charge density and its derivati e with respect to potential. PSSern apportions the charge (density returned by RhoOfP times the element volume) to the nodes of the element as if the charge was concentrated at a point in the middle of the element. The derivative is treated as element-centered.

3.6.1. Laplacian Space Charge

The Laplace space charge option solves Laplace's equation,

$$-\nabla^2\phi = 0$$

i.e., charge exists only on object surfaces and external boundaries, as determined by the boundary conditions. "Space charge" iterations may still be required, however, due to the treatment of surface electric fields.

3.6.2. Linear Space Charge

The Linear space charge option solves the Helmholtz or Debye-Huckel equation

$$-\nabla^2\phi = -\phi/\lambda^2$$

The value of λ is the maximum of the plasma Debye length (divided by the square root of density depletion due to geometric shadowing), and the local mesh spacing divided by the DEBLIM parameter (which defaults to 2).

3.6.3. Nonlinear Space Charge

Nonlinear space charge is calculated using the NASCAP/LEO formula:

$$-\nabla^2\phi = \rho/\varepsilon_0$$
$$\rho/\varepsilon_0 = -(\phi/\lambda_D^2)\,(1 + |\phi/\theta|\,C(\phi,E))\,/\,(1 + (4\pi)^{1/2}\,|\phi/\theta|^{3/2})$$
$$C(\phi,E) = |\theta/\phi|\;[(R_{sh}/r)^2 - 1]$$
$$(R_{sh}/r)^2 = 2.29\,|E\lambda_D/\theta|^{1.262}\,|\theta/\phi|^{.509}$$

ρ = space charge [coul-m^{-1}]
$\varepsilon_0 = 8.854\times10^{-12}$ [farad-m^{-1}]
λ_D = plasma Debye length [m]
θ = plasma temperature [eV]
ϕ = *local space potential [volts]*
E = local space electric field [volts-m^{-1}]

where the last equation (analytic focusing) comes from fitting a finite-temperature spherical (Langmuir-Blodgett) sheath. If analytic convergence is turned off (by the CONV input parameter) $C(\phi,E)$ is set to zero. The value of λ_D is modified in accordance with the mesh spacing as described for linear screening (above).

3.6.4. Frozen Ion

The "Frozen Ion" treatment is intended for short timescale (typically a few microseconds or less) problems for which it is a good approximation to assume that electrons are in barometric equilibrium, but ions have not moved. The space charge function depends on the mesh-dependent potential, ϕ_1, which satisfies

$$1 - \exp(\phi_1/\theta) = -(\lambda/D)^2\,(\phi_1/\theta)$$

where D is the local mesh spacing divided by the DEBLIM parameter. (ϕ_1 is zero for $D\leq\lambda$.) The space charge, ρ/ε_0, is then given by

$$\begin{array}{ll} (\phi_1/D^2)(1-\exp(\phi/\phi_1)) & \phi\geq0 \\ -\phi/D^2 & 0\geq\phi\geq\phi_1 \\ -\phi_1/D^2 + (\theta/\lambda^2)(\exp(\phi_1/\theta)-\exp(\phi/\theta)) & \phi\leq\phi_1 \end{array}$$

3.6.5. Tracked Ions

This algorithm is used for timescales on which it is practical to treat ion motion, but electrons are considered in barometric equilibrium. The ion density has been stored in the RHO_Ion array by the Tracker module, and the electron charge density must be added. Subroutine RhoOfP returns the electron charge density, ρ/ϵ_0, as

$$\begin{array}{ll} -(\phi+\theta(L/\lambda)^2)/L^2 & \phi\geq 0 \\ -(\theta/\lambda^2)\exp(\phi\lambda^2/\theta L^2) & \phi\leq 0 \\ L = \max(\lambda, D) & \end{array}$$

3.6.6. Explicit PIC

For this option, it is assumed that the Tracker module has stored both the electron and ion charge densities in the RHO_Ion array.

3.6.7. Sheath Wake Densities

This option is intended for problems in which the "GI" density array has been calculated using shadowing by a positive potential contour (rather than by the object). The module which performs this calculation is not completely developed. The section of RhoOfP which treats this case is intended to blend between regions with accelerated electrons, quasi-neutral regions, and electron rich wake regions. We do not attempt to explain the logic here, but refer to interested reader to the RhoOfP subroutine. (Shadowing is not appropriate to large negative sheaths. A variant of the "Tracked Ion" method should be developed for such cases.)

3.6.8. Special Space Charge Formulation

The option is available to call a user-written routine, RSpecl, for a developmental space charge option. One array which might be used in such an option is RHO_Elec, the element-centered electron charge density, calculated by the Tracker module by tracking full electron orbits in existing potentials.

3.7. Selected Subroutine Descriptions

There are no subroutine descriptions supplied at this time. Just to provide something, here is a description of the "element table", LTBL, which is written into the DataBase for each grid. For each volume element the LTBL entry is

<0	Element is filled or belongs to child grid.
0	Element is empty.
>0	Special element index number.

4. DynaPAC Example - Part I

This example originally appeared in the Interim Report, SSS-DPR-90-11973, 30 September 1990.

4.1. Introduction to Using DynaPAC

This chapter takes you through a sample calculation using many of the DynaPAC modules. In this example, you will use DynaPAC to calculate the potential about a charged cube. The calculation proceeds as follows:

1. Use PATRAN to define a cube with a side of 20 centimeters. You will also create an auxiliary material file that defines the surface material to be aluminum.

2. Use GridTool to define a grid about the cube with one level of subdivision.

3. Use PatDyn to initialize the DynaPAC data base and place in it the needed geometrical information.

4. Use the "main" module to establish an initial potential (IPS) of -1000 volts on the cube and to create an input file for the potential solver. Among the input specifications are the plasma parameters for which you will use a density of 1 ¥ 1011 m-3 and a temperature of 0.1 eV.

5. Run the potential solver.

6. Use the data scanner to print the potentials and electric fields on the cube surface and in the space near the cube and to plot the potentials in space.

4.1.1 Using PATRAN to Define the Cube.

Figure 4.1.1 shows the PATRAN session file to define the cube.

Lines 1-2	Sign on to PATRAN, requesting a new datafile.
Lines 3-6	Create a cube of the required dimensions.
Lines 7-8	Define "PATCH's" on the faces of the cube.
Line 9	Create an element of material 1 on each PATCH.
Lines 10-16	Merge the equivalent nodes and compact the node numbers.
Line 17	Check for free boundaries.
Lines 18-19	Display normals.
Lines 20-21	Correct normals to be consistently outward.
Lines 22-28	Write the neutral file.
Line 29	Exit PATRAN.

```
 1    GO
 2    1
 3    PRIM,CUBE.BRICK
 4    1
 5    0.2.0.2,0.2
 6    5
 7    PRIM,CUBE,EV
 8    S
 9    MESH,P1T6,QUAD/4/1,N,1/1/1/1,-1
10    EQUIVALENCE
11    N
12    2
13    1
14    Y
15    6
16    2
17    VBOUND
18    VNORM
19    1
20    3
21    1
22    INTERFACE
23    1
24    1
25    1
26    CUBE FOR SAMPLE PROBLEM
27    N
28    N
29    STOP
```

Figure 4.1.1 PATRAN session file to define the cube.

Finally, rename the neutral file with the command

```
mv patran.out.1 cube.neu
```

and create the material file "cube.mat" containing the single line

```
1 Aluminum
```

4.2 Using Grid Tool

1. Issue the "GridTool" command to run the GridTool module of DynaPAC.

2. Choose the "Files" option in order to specify the file prefix. Change it from "fort" to "cube."

3 Accept the default of a PATRAN neutral file with units of meters.

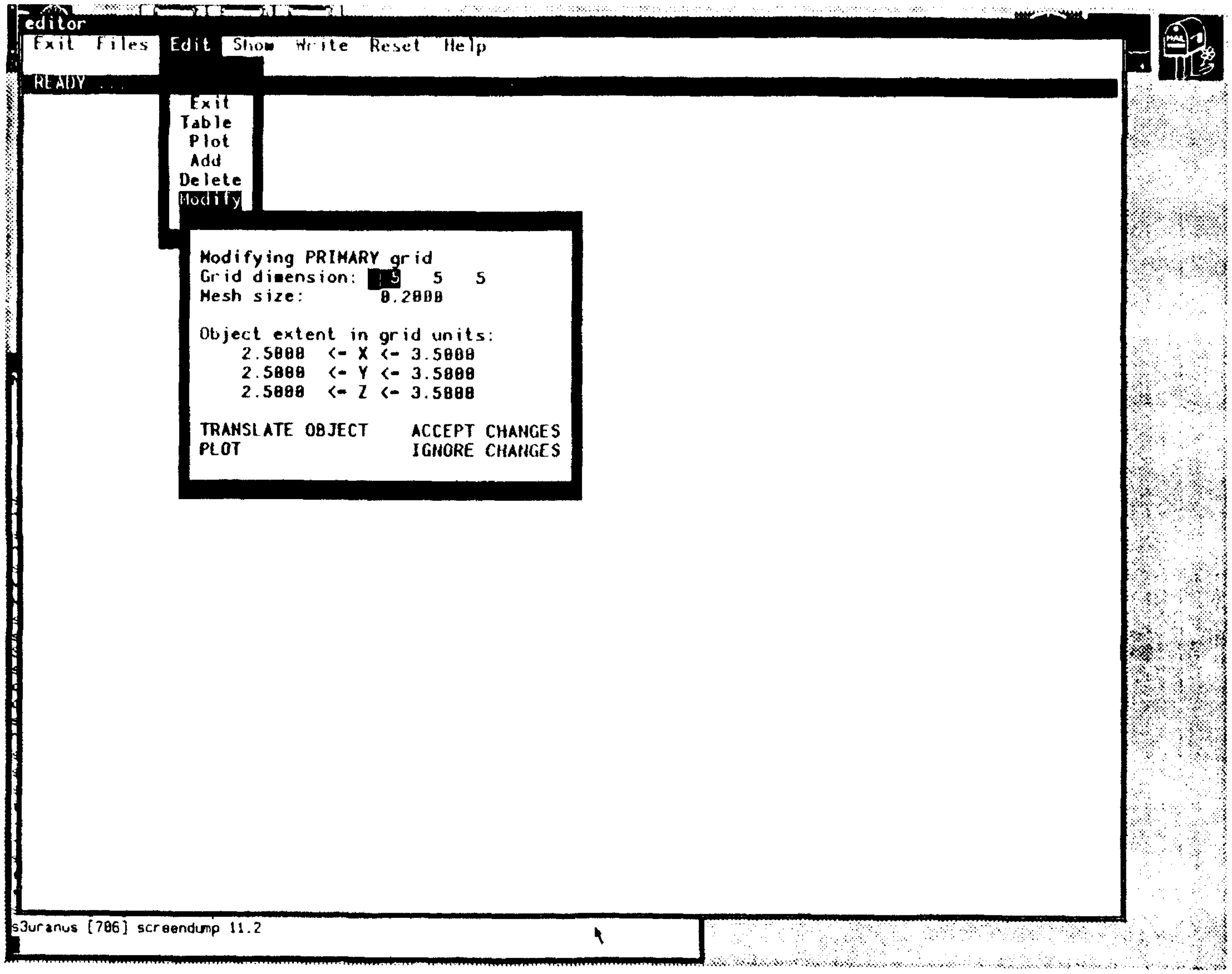
4. Choose the "Edit" option to change the grid parameters. Start with the only existing grid, grid #1, and choose the "Modify" screen. The console now appears as shown in figure 11.2.

5. Alter the grid dimensions to 13 x 13 x 13 and the mesh size to 0.6 meters, and accept those changes.

6. Next, choose the "Add" option from the pull-down menu in order to place a subdivided grid around the cube. The screen now appears as in figure 11.3. Change the subdivision ratio to 3, and the grid limits to run from 5 to 9 in each direction. Accept the grid.

7. Choose the "Plot" option from the pull-down menu. Using the default parameters, make and show the plot, which appears as in figure 11.4.

8. Exit the "Edit" menu, select the "Write" option to write the grid definition file, "cube.grd," and exit GridTool.



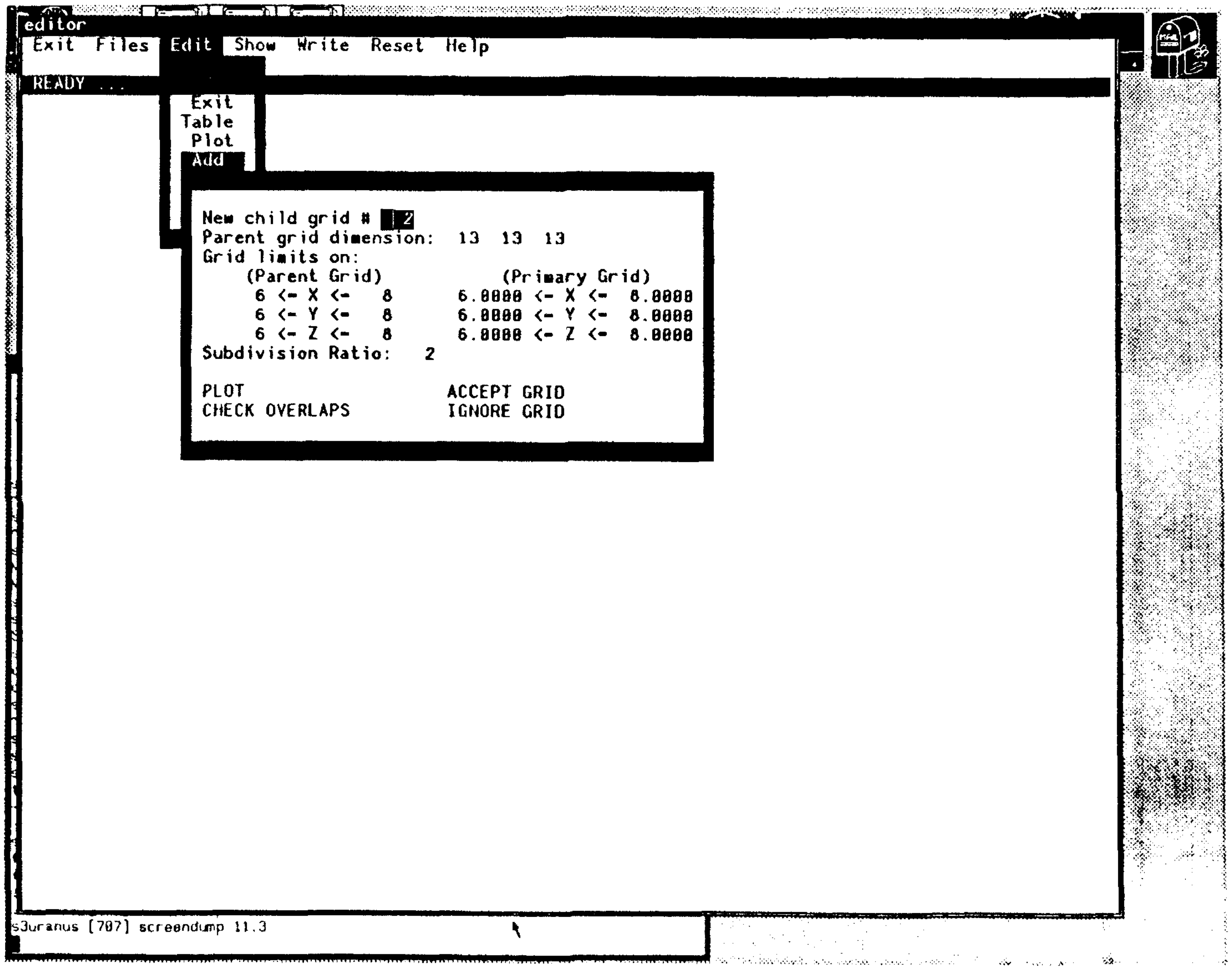
```
Modifying PRIMARY grid
Grid dimension:  5    5    5
Mesh size:          0.2000

Object extent in grid units:
    2.5000  <- X <- 3.5000
    2.5000  <- Y <- 3.5000
    2.5000  <- Z <- 3.5000

TRANSLATE OBJECT     ACCEPT CHANGES
PLOT                 IGNORE CHANGES
```

```
s3uranus [706] screendump 11.2
```

Figure 4.2 GridTool screen for the Edit/Modify option.



```
New child grid # 2
Parent grid dimension:  13  13  13
Grid limits on:
    (Parent Grid)            (Primary Grid)
      6 <- X <-   8       6.0000 <- X <-  8.0000
      6 <- Y <-   8       6.0000 <- Y <-  8.0000
      6 <- Z <-   8       6.0000 <- Z <-  8.0000
Subdivision Ratio:   2

PLOT                     ACCEPT GRID
CHECK OVERLAPS           IGNORE GRID
```

```
s3uranus [707] screendump 11.3
```

Figure 4.3 GridTool screen for the Edit/Add option.

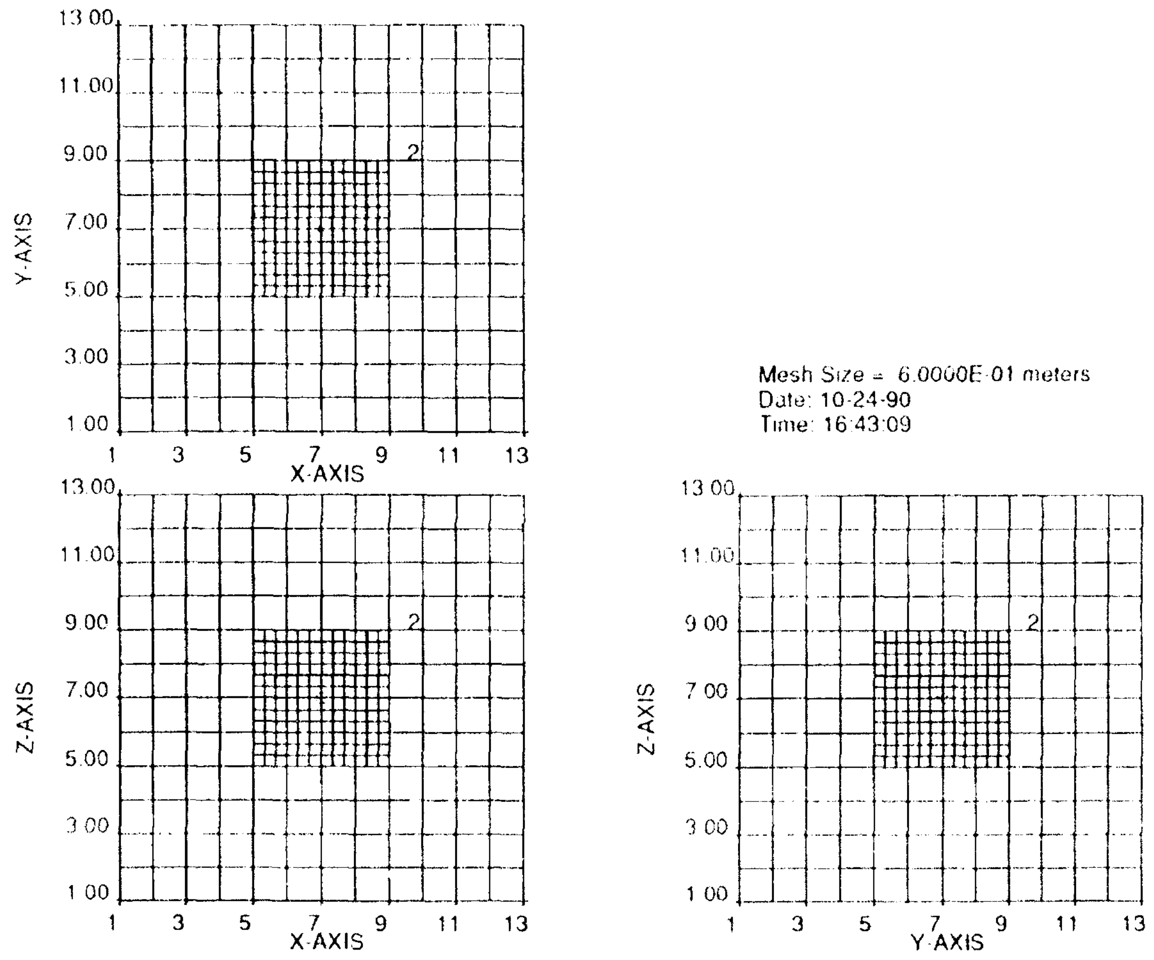


Figure 4.4 GridTool plot for the sample problem.

4.3 Running PatDyn

PatDyn can now be run with the command

```
PatDyn_SUN4 < patdyn.in > patdyn.out &
```

The first line of the input file, "patdyn.in", uses the PREFIX keyword to set the file prefix. Normally, no other lines are needed, so, in this case, "patdyn.in" appears as

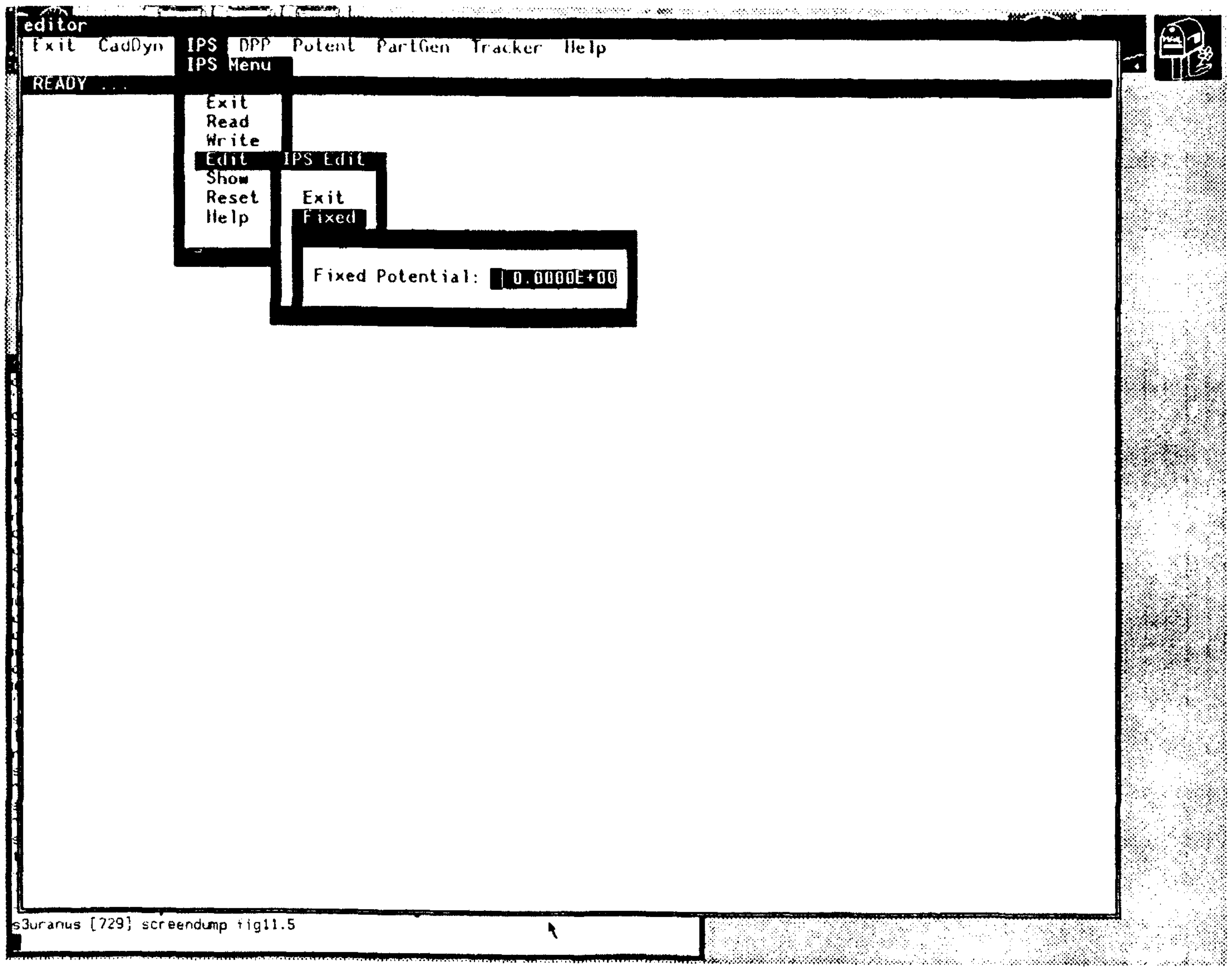
```
PREFIX cube
END
```

(The input file, "patdyn.in", can also be built by the DynaPre module.) The output file, "patdyn.out", repeats the mesh specification information and contains information on any possible errors that may have been detected in processing the object. It concludes by giving the properties of the materials, although these are not currently used by DynaPAC.

4.4 Using DynaPre to Prepare to Solve For Potentials

Enter the DynaPre module by issuing the command "DynaPre". The first task is to specify a potential of -1000 volts on each face of the cube. Choose the "IPS" option from the top menu, and choose the "Edit" submenu.

The next screen offers a range of conductors and materials to which initial conditions are to be applied. By default, this is the entire range available. Accept this choice. A pull-down menu now appears offering a choice of the type of boundary condition to be applied. Choose "Fixed". The screen now appears as shown in figure 4.5. Change the specified potential to -1000. You are instantly returned to the main IPS menu. Write out the initial conditions, and exit IPS.



Fixed Potential: 0.0000E+00

```
s3uranus [729] screendump fig11.5
```

Figure 4.5 DynaPre screen for the IPS/Fixed option.

Now choose the "Potent" option from the top menu and the "Edit Script" option from the pull-down menu. You will be presented with the default potential solver input options, as shown in figure 11.6. Change the "Comment" to "Cube Sample Problem" and toggle the "Problem type" to "NON_LINEAR". Note that the density and temperature are already properly set. Exit this screen and choose the "Make Script" option from the pull-down menu. This will write a fully commented input file in "ps_in". Exit the pull-down menu and the "DynaPre" module, and inspect this file.

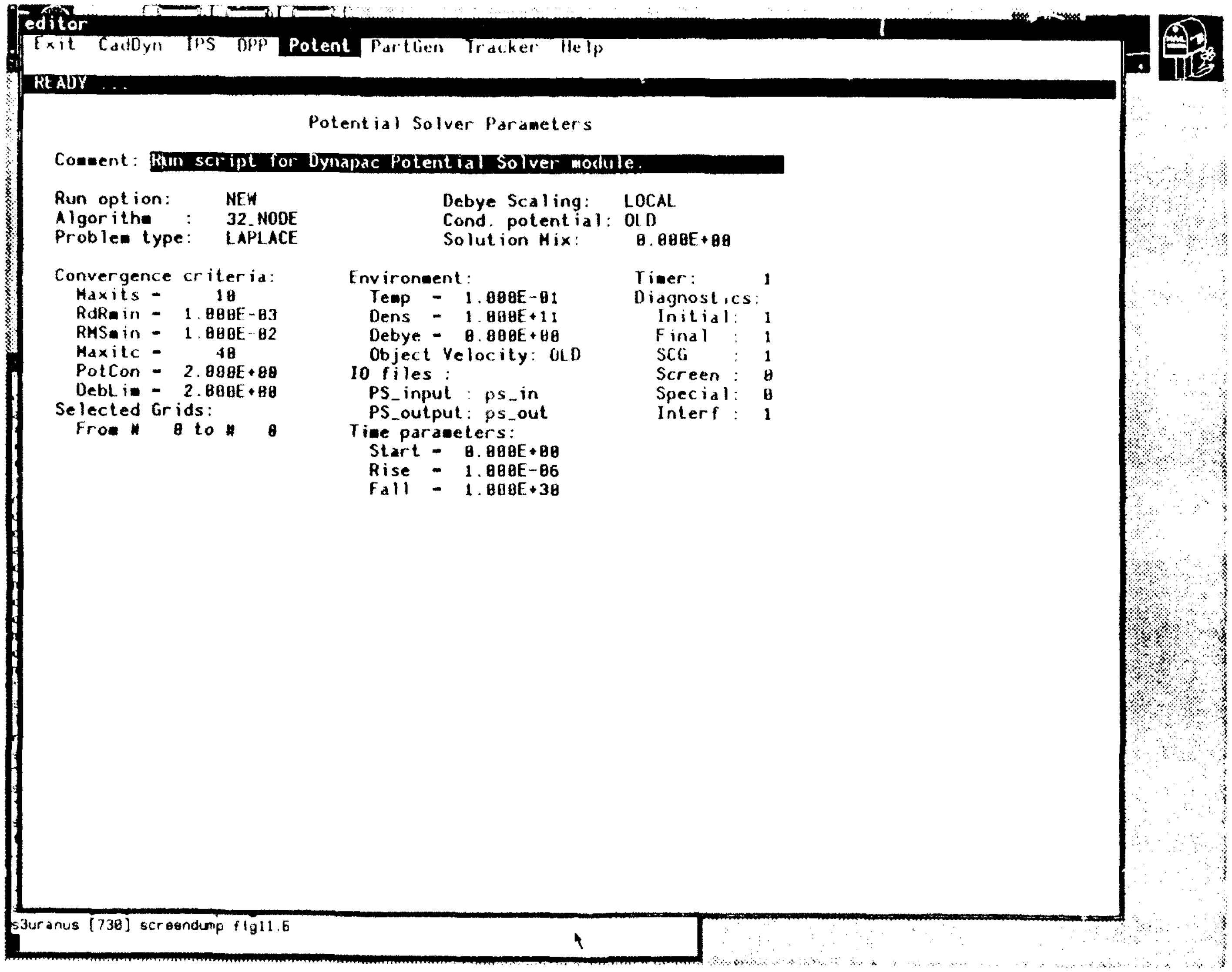
4.5. Running the Potential Solver

Run the potential solver using the command

```
Potent_SUN4 < ps_in > ps_out &
```

The output file contains a reiteration of the input, problem, and grid parameters, and details the convergence of the potential solver.

Before proceeding, note the "Fort" and "PS" prefix files. "Fort.DP" contains object surface information and various miscellaneous information about the calculation. "Fort.BS" contains information about the bounding surfaces of the special elements and is needed to calculate electric fields in special elements or to plot potentials in space. "Fort.ME02" contains the potential solver matrices for special elements in grid 2. There will be one such file for each grid containing special elements. "PS.DP" contains information generated by the potential solver. The ".HI" files contain the data base specifications that characterize the data entities in the other files. If you wish to save results from calculations done with several different parameters, you need save only the ".DP" files; the remaining files are not changed.



```
                         Potential Solver Parameters

 Comment: Run script for Dynapac Potential Solver module.

 Run option:     NEW                Debye Scaling:   LOCAL
 Algorithm   :   32_NODE            Cond. potential: OLD
 Problem type:   LAPLACE            Solution Mix:     0.000E+00

 Convergence criteria:      Environment:                Timer:        1
   Maxits -      10           Temp  -  1.000E-01        Diagnostics:
   RdRmin -  1.000E-03        Dens  -  1.000E+11          Initial:  1
   RMSmin -  1.000E-02        Debye -  0.000E+00          Final   :  1
   Maxitc -     40            Object Velocity: OLD        SCG     :  1
   PotCon -  2.000E+00      IO files :                    Screen  :  0
   DebLim -  2.000E+00        PS_input  : ps_in           Special:  0
 Selected Grids:              PS_output: ps_out           Interf  :  1
   From #    0 to #    0    Time parameters:
                              Start -  0.000E+00
                              Rise  -  1.000E-06
                              Fall  -  1.000E+30
```

```
s3uranus [730] screendump fig11.6
```

Figure 4.6 DynaPre screen for the POTENT/Edit Script option.

4.6 Using the Data Scanner

1. Issue the command "Scanner" to run the data scanner. Choose "Print" from the top menu, and "Data Info" from the pull-down menu. Change the "Working Grid" to 2, and exit the "Data Info" screen.

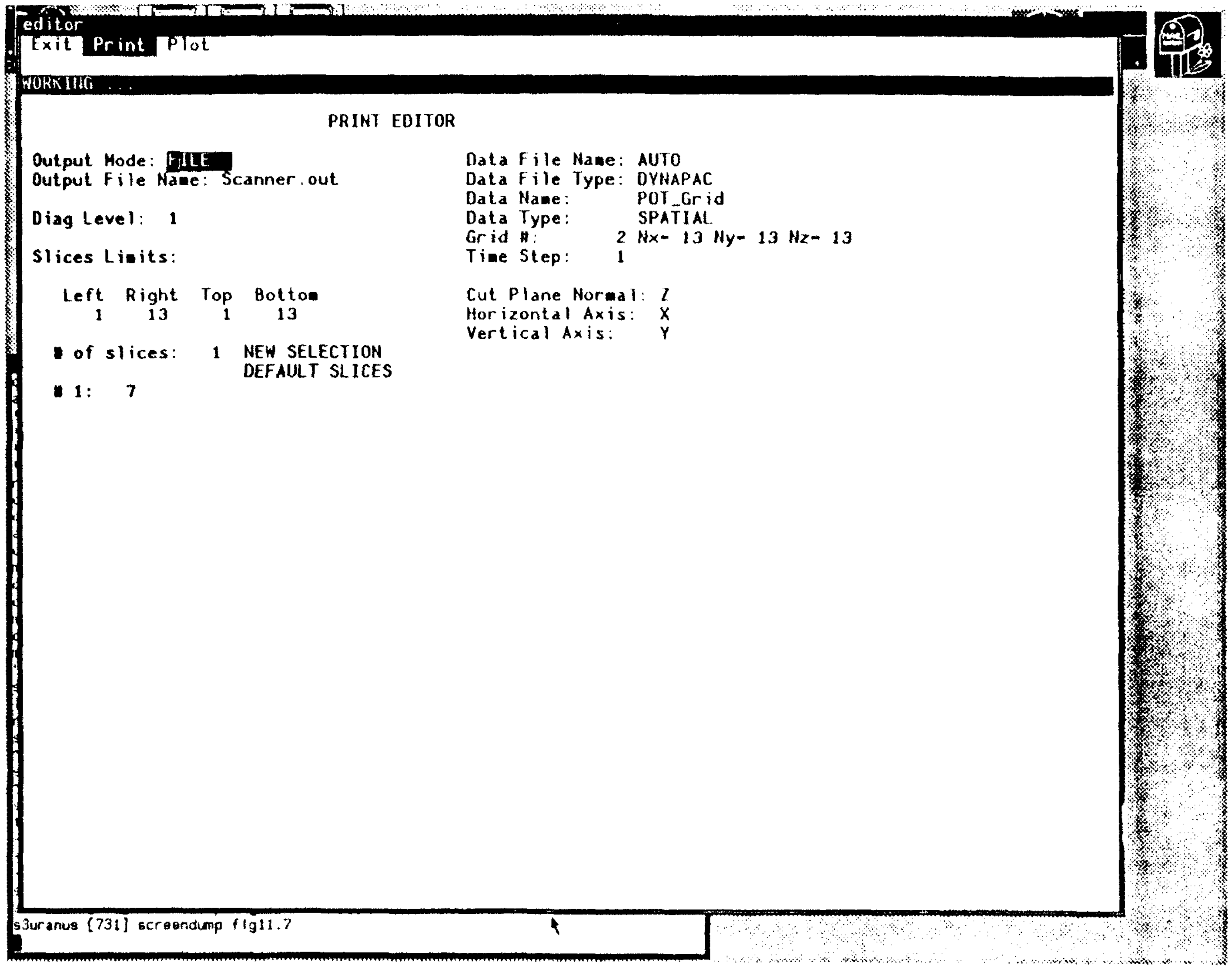
2. Choose "Edit Print" from the pull-down menu. You will now see the screen shown in figure 11.7. You have the option of printing to the screen or to a named file (which can later be edited). Toggle the "Output Mode" to "SCREEN". Also, change the "Right" and "Bottom" limits to 9, and the "Left" and "Top" limits to 3. Exit the screen.

3. Choose "Make Print" from the pull-down menu. The potentials and potential gradients at the node points of the z=7 (middle) grid plane are printed. The potential (volts) for each point is at the intersection of the appropriate row and column. The potential gradient (volts/meter) in the y-direction is printed above the potential, the x-direction gradient is printed to the right. The entries at (7,7) are all zero, as this point is inside the cube. Note that the potentials and fields are consistent and that there is a slight asymmetry due to asymmetry in the element bounding surfaces.

4. Return to the pull-down menu and again choose "Data Info". Specify "POT_Surf" as the "Data Name". Review the "Edit Print" screen (it should be correct) and select "Make Print". The potentials (-1000 volts) and normal electric fields (-5000 volts/meter) will appear on the screen. Again, the field is slightly asymmetric due to asymmetry in the bounding surfaces. Exit the "Print" module.

5. Select "Plot" from the top menu. Review the "Data Info" screen, and choose "Edit Plot". Figure 11.8 shows the "Edit Plot" screen.

6. Choose "GRID LIMITS" and change the limits to plot from 3 to 9 in each direction (this time in units of the outermost grid). Exit the screen.



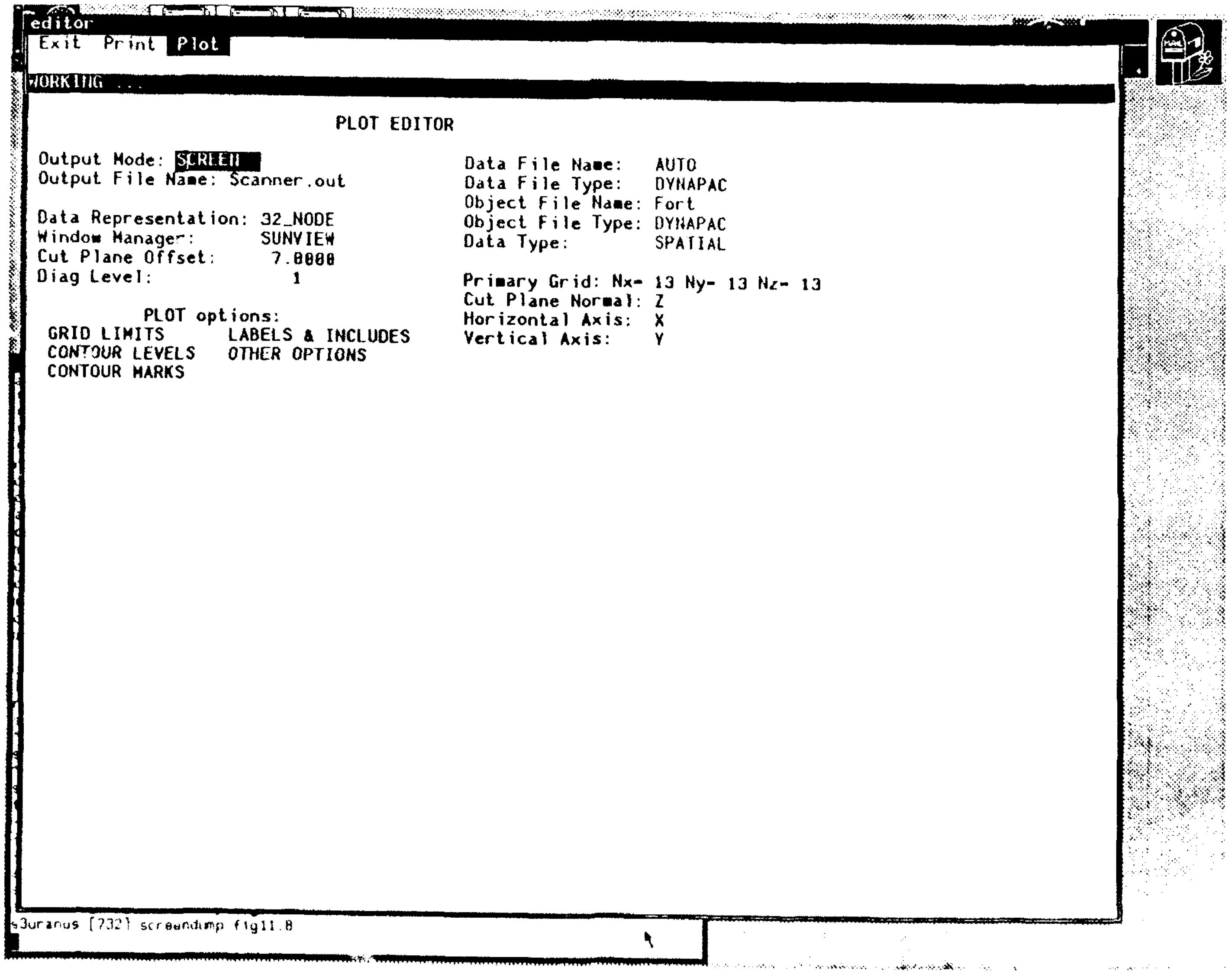
```
                         PRINT EDITOR

Output Mode: FILE                        Data File Name: AUTO
Output File Name: Scanner.out            Data File Type: DYNAPAC
                                         Data Name:      POT_Grid
Diag Level:  1                           Data Type:      SPATIAL
                                         Grid #:        2 Nx= 13 Ny= 13 Nz= 13
Slices Limits:                           Time Step:     1

   Left  Right  Top  Bottom              Cut Plane Normal: Z
     1    13     1     13                Horizontal Axis:  X
                                         Vertical Axis:    Y
  # of slices:   1  NEW SELECTION
                    DEFAULT SLICES
  # 1:   7
```

```
s3uranus [731] screendump fig11.7
```

Figure 4.7 Scanner screen for the Print/Edit Print option.



```
                     PLOT EDITOR

Output Mode: SCREEN                        Data File Name:   AUTO
Output File Name: Scanner.out              Data File Type:   DYNAPAC
                                           Object File Name: Fort
Data Representation: 32_NODE               Object File Type: DYNAPAC
Window Manager:      SUNVIEW               Data Type:        SPATIAL
Cut Plane Offset:     7.0000
Diag Level:             1                  Primary Grid: Nx= 13 Ny= 13 Nz= 13
                                           Cut Plane Normal: Z
           PLOT options:                   Horizontal Axis:  X
 GRID LIMITS      LABELS & INCLUDES        Vertical Axis:    Y
 CONTOUR LEVELS   OTHER OPTIONS
 CONTOUR MARKS

s3uranus [732] screendump fig11.8
```

Figure 4.8 Scanner screen for the Plot/Edit Plot option.

7. Choose "CONTOUR LEVELS". Change the number of contours to 15. The contour levels are recalculated instantly, and the screen is updated to show 15 levels. Change contour levels 12-15 to -30, -10, -1, and 0 volts, respectively. Exit the screen.

8. Choose "CONTOUR MARKS". Change "Number of Marks" to 2. The screen is redrawn to show 2 marked contours. Change the second marked contour value to -30 volts and the mark to "3". Exit the screen.

9. Choose "LABELS AND INCLUDES". Change the "Title" to "Cube Example". Exit the screen.

10. Select "OTHER OPTIONS". If you are on a color terminal, toggle that option to "YES".

11. Return to the pull-down menu, and select "Make Plot" and "Show Plot". A gray-scale version of the contour plot appears in figure 11.9. A black-and-white version of the contour plot appears in figure 11.10. Hit <RET> to go back to the menu. Exit "Plot" menu, and then exit Scanner.

Note the presence of the files "fort.2" and "Scanner.opt". The plot data is contained in "fort.2" and can be displayed or printed using any of the DynaPAC graphical interfaces. "Scanner.opt" stores the options used when running the data scanner, which will be restored in the next Scanner run in this directory.

4.7 Summary

This example has taken you from defining an object through calculating and displaying the potentials in the surrounding space.

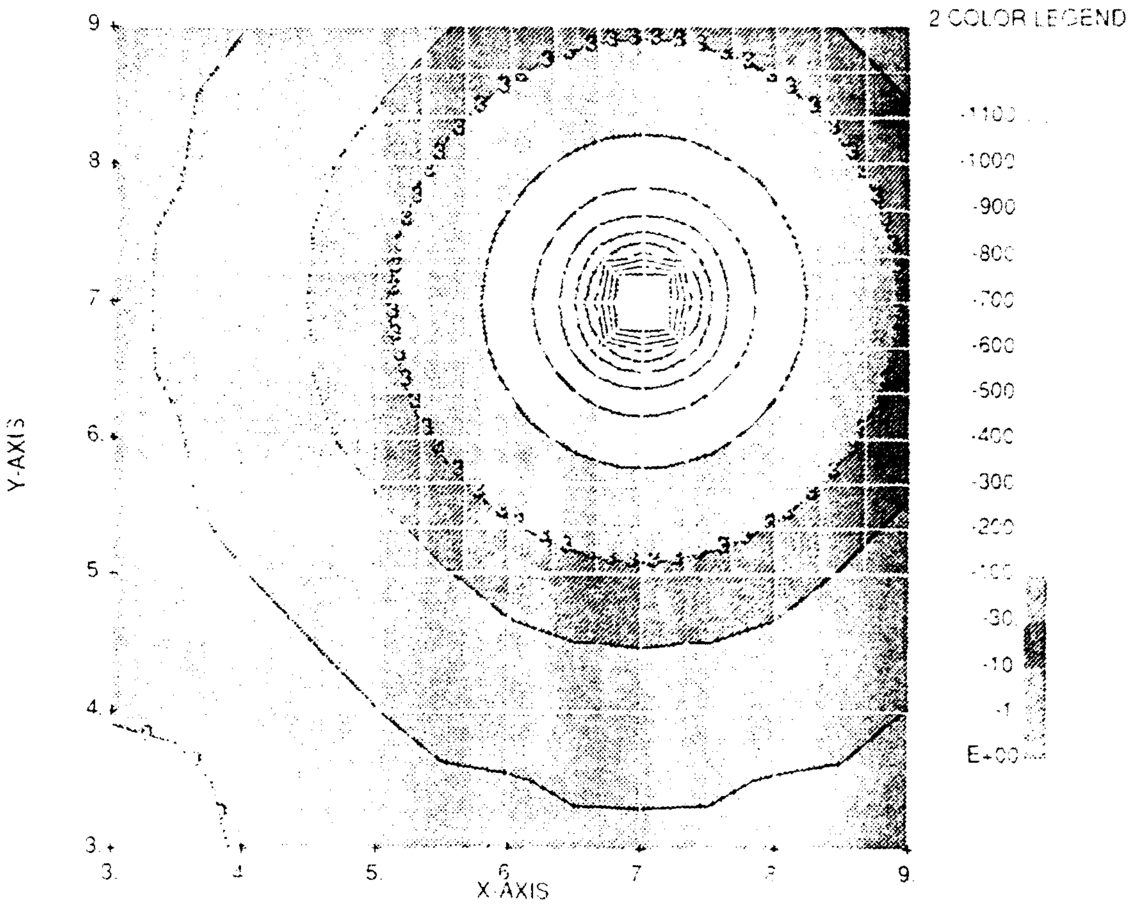


Figure 4.9 Contour plot of potentials for sample problem, gray-scale version.

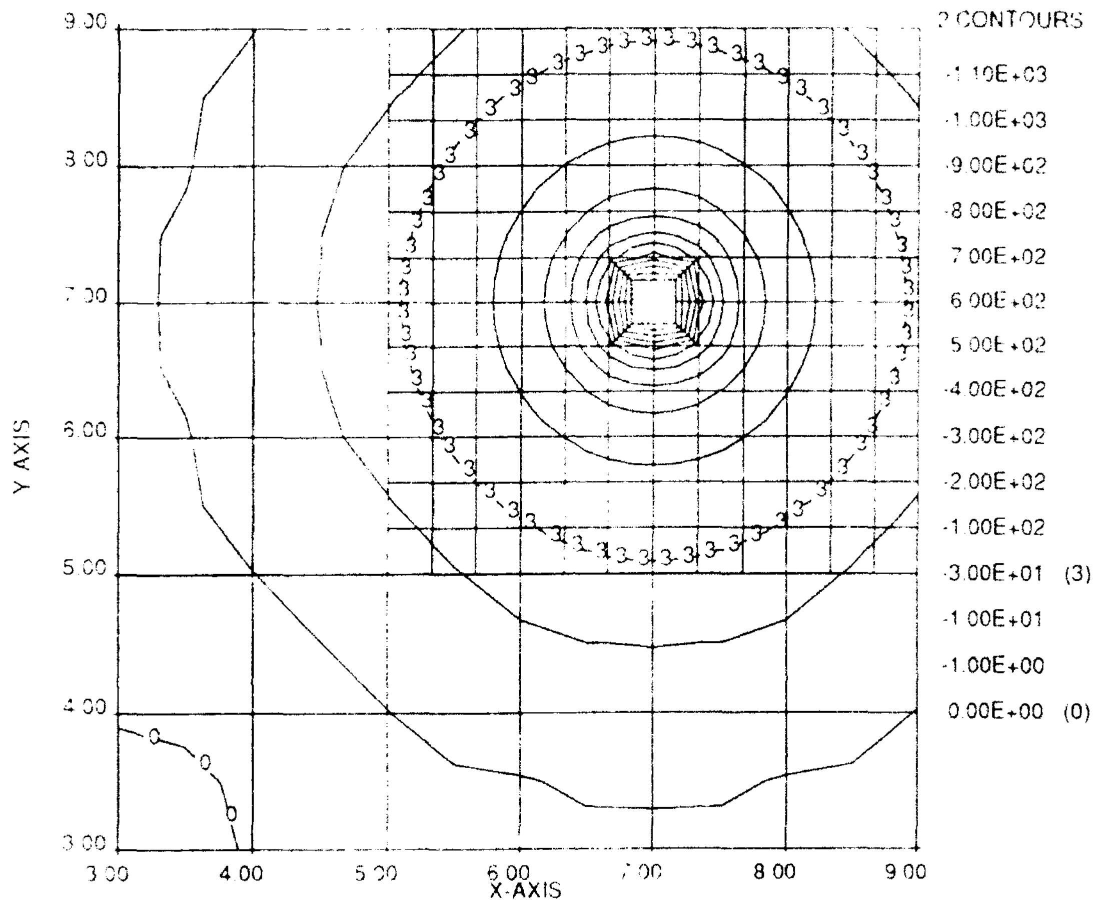


Figure 4.10 Contour plot of potentials for sample problem, black-and-white version.

5. DynaPAC Example - Part II

This example originally appeared in the Quarterly Report, SSS-DPR-91-12658, July 1991.

5. A Sample Calculation Using DynaPAC

5.1 General

The goal of this calculation is to: (1) calculate the biased floating potential about an object consisting of two disjoint, unequal size cubes; (2) examine some particle trajectories from the sheath (ions as well as electrons).

The calculation proceeds as follows:

1. Use PATRAN to define two cubes having sides of 20 cm and 30 cm, respectively, and being 20 cm apart. The material file will define the surface materials to be aluminum and silver, respectively.

2. Use GridTool to define two nested grids about the object.

3. Use PatDyn to initialize the DynaPAC database and place in it the needed geometrical information.

4. Use the "DynaPre" module to establish a biased potential set where the small cube's potential is initialized to -10 volts and the large cube's potential is biased +10 volts relative to the small cube's. Also create an input file for the potential solver.

5. Run the potential solver.

6. Use the data scanner to print the potentials and electric fields on the object surfaces and in the space near the cubes and to plot the potentials in space.

7. Use "DynaPre" again to create an input file for the particle generator. Then run particle generator (PartGen) to generate sheath particles.

8. Use "DynaPre" again to create two input files for the particle tracker (Tracker). One input file tells Tracker to plot the particles. The other input file tells Tracker to track those particles and generate a trajectory plot. Then run Tracker twice with appropriate input file as described above.

5.2 Parameter

The parameters used in the calculations will be:

Plasma density	1×10^{11}	m^{-3}
Plasma temperature	0.1	eV
Plasma species	O^+, e^-	
Magnetic field	4×10^{-5}	tesla

5.3 Using PATRAN to define the cube.

Refer to sample 1 for how to define an object using PATRAN. Define the object (two cubes). Finally, at the UNIX prompt, rename the neutral file with the command

```
%mv patran.out.1 2Cubes.neu
```

and create the material file "2Cubes.mat" containing two lines

```
1 Aluminum
2 Silver
```

5.4 Using GridTool

1. Issue the "GridTool" command to run the GridTool module of DynaPAC.

2. Choose the "Files" option in order to specify the file prefix. Change it from "fort" to "2Cubes."

3. Accept the default of a PATRAN neutral file with units of meters.

4. Choose the "Edit" option to change the grid parameters. Start with the only existing grid, grid 1, and choose the "Modify" screen. The console now appears as shown in figure 5.1.

5. Alter the grid dimensions to 13 x 13 x 13 and the mesh size to 0.16 meters and accept those changes.

6. Next, choose the "Add" option from the pull-down menu in order to place a subdivided grid around the two cubes. The screen now appears as in figure 5.2. Change the grid limits to run from 4 to 10 in the X direction and from 5 to 9 in the Y and Z directions. Accept the grid.

7. *Choose the "Plot" option from the pull-down menu.* Using the default parameters, toggle "PLOTTER" to your favorite plot- reader. Make and show the plot, which appears as in figure 5.3.

8. Exit the "Edit" menu, select the "Write" option to write the grid definition file, "2Cubes.grd," and exit GridTool.

5.5 Running PatDyn

PatDyn can now be run with the command

```
%PatDyn_SUN4 < patdyn.in > patdyn.out &
```

The first line of the input file, "patdyn.in," uses the PREFIX keyword to set the file prefix. Normally, no other lines are needed, so in this case, "patdyn.in" appears as

```
PREFIX 2Cubes
END
```

(The input file, "patdyn.in," can also be built by the DynaPre module.) The output file, "patdyn.out," repeats the mesh specification information and contains information on any possible errors that may have been detected in processing the object. It concludes by giving the properties of the materials.

5.6 Using DynaPre to prepare to solve for potentials

Enter the DynaPre module by issuing the command "DynaPre." Enter "2Cubes" at the "File Prefix:" prompt. The task is to specify a biased floating conductor set where conductor 1 (small cube) gets initialized to -10 volts and conductor 2 (large cube) is biased +10 volts relative to conductor 1.

Choose the "IPS" option from the top menu; then choose the "Conductors" submenu, and choose "Set" option.

The next screen shows a list of all conductors defaulted to "Fixed Potential." Toggle conductor 1 type to "Biased Potential" and set its "Value" to -10 volts (leaving "Biased From" to be "0" since this is the lowest conductor number in the set). Toggle conductor 2 type to "Biased Potential," set its "Value" to +10 volts, and set "Biased From" to "1." The screen now appears as in figure 5.4.

Now, go back to the main IPS menu (by hitting <ESC> twice). Select "Write" to write out the initial conditions, and exit IPS. An ascii file named "ips.out" is also written out for diagnostic purposes.

Now choose the "Potent" option from the top menu, and the "Edit Script" option from the pull-down menu. You will be presented with the default potential solver input options, as shown in figure 5.5. Change the "Comment" to "Biasing Cubes Sample Problem," and toggle the "Problem type" to "NON_LINEAR." Note that the density and temperature are already properly set. Exit this screen and choose the "Make Script" option from the pull-down menu. This will write a fully commented input file in "ps_in." Exit the pull-down menu and the "DynaPre" module, and inspect this file.

5.7 Running the potential solver

Run the potential solver using the command

```
%Potent_SUN4 < ps_in > ps_out &
```

The output file contains a reiteration of the input, problem, and grid parameters, and details the convergence of the potential solver.

Before proceeding, note the 2Cubes.* files created so far.

2Cubes.DP	Main data file containing most of information about the calculation.
2Cubes.BS	Contains information about the bounding surfaces of the special elements and is needed to calculate electric fields in special elements or to plot potentials in space.
2Cubes.ME02	Contains the potential solver matrices for special elements in grid no. 2. There will be one such file for each grid containing special elements.
2Cubes.HI	Contains the database specifications that characterize the data entities in the other files.

If you wish to save results from calculations done with several different parameters, you need save only the ".DP" file__the remaining files are not changed.

5.8 Using the Data Scanner

1. Issue the command "Scanner" to run the data scanner. Again, enter "2Cubes" at the "File Prefix" prompt.

2. Select "Print" from the top menu and choose "Data Info." Change "Data Name" to "POT_Surf" and exit this form.

3. Select "Edit Print" and toggle "Output Mode" from "FILE" to "SCREEN." Exit this form and select "Make Print." The screen will display all surface potentials and fields as in figure 5.6. Note that the total net charge on the two cubes is almost zero. Exit this screen and go back to main Scanner menu.

4. Select "Plot" from the top menu and choose "Edit Plot." Figure 5.7 shows the "Edit Plot" screen.

5. Toggle "Window Manager" to the most appropriate one. Toggle "Plotter" to your favorite "fort.2 plot-reader."

6. Choose "GRID LIMITS." Change limits to run from 2 to 12 horizontally and from 3 to 11 vertically. Exit this screen.

7. Choose "CONTOUR LEVELS." Change the number of contours to 14. The contour levels are recalculated instantly, and the screen is updated to show 14 levels. Change contour levels 13 and 14 to +0.2, and -0.2 volts, respectively. Exit the screen.

8. Choose "CONTOUR MARKS." Change "Number of Marks" to 3. The screen is redrawn to show 3 marked contours. Change the second marked contour value to -.2 volts and the mark to "-". Change the third marked contour value to +.2 volts, and the mark to "+". Exit the screen.

9. Choose "LABELS AND INCLUDES." Change the "Title" to "Biasing Cubes Example." Exit the screen.

10. Select "OTHER OPTIONS." If you are on a color terminal, toggle that option to "YES." Exit this screen.

11. Return to the pull-down menu, and select "Make Plot" and then "Show Plot." A black-and-white version of the contour plot appears in figure 5.8. Hit <RET> to go back to the menu. Exit "Plot" menu, and then exit Scanner.

Note the presence of the files "fort.2" and "2Cubes.scan." The plot data is contained in "fort.2" and can be displayed or printed using any of the DynaPAC graphical interfaces. "2Cubes.scan" stores the options used when running the data scanner, which will be restored in the next Scanner run in this directory.

5.9 Running Particle Generator (PartGen)

Issue the command "DynaPre" to generate input file to module PartGen. From the main menu, choose "PartGen" and then select "Edit Script." The screen will appear as in figure 5.9. Change the "Comment" to "Biasing Cubes Sample Problem." Toggle "Particle Type" to "CONTOUR." A pop-up form will come up showing the default cut-plane orientation. Change "Contour Potential" to "0.1" volts, and change "Cut Offset" to "7.0" (middle Z-plane). Exit this form. Now, change the "Number of species" to "2." The screen will change instantly to show two species (O^+ and e^- as defaults). Exit this form. Select "Make Script" from the menu to write out file "pg_in." Exit "PartGen" and exit "DynaPre". Inspect the file "pg_in." At the UNIX prompt, type:

```
%PartGen_SUN4 < pg_in > pg_out &
```

The output file, "pg_out," echoes the input file "pg_in," shows general problem and grid parameters, and details the particles generated.

5.10 Running Particle Tracker (Tracker)

Run "DynaPre" again to generate input file to module Tracker. Choose "Tracker" from the main menu and then select "Edit Script." The screen will appear as in figure 5.10.

Change the "Comment" to "Biasing Cubes Sample Problem." Change "Title" to "Plotting Particles at sheath=0.1 volts." Change "Number of Species" to "2" to specify two different species (species 1 and species 2). Toggle "Process" to "PLOT_PARTICLES." A new form is popped up. Change "Cut Offset" to "7.0" (middle Z-plane). Exit this form The screen will change instantly to show defaults corresponding to the selected option "PLOT_PARTICLES." Change "Plot Limit" to run from 2 to 12 in X direction and from 3 to 11 in Y direction. The screen now appears as in figure 5.11. Exit this form and select "Make Script" to write out file "tr_plot_in."

Select "Edit Script" again to return to "Edit" form. We are going to create yet another input file to Tracker. Change "Title" to "Tracking Particles from sheath=0.2 volts." Toggle "Process" to "TRAJECTORIES" and accept the projection plane. The screen will change to show defaults corresponding to the selected option "TRAJECTORIES." Change "B Field" to be "4.e-5 tesla" in "Z" direction. Increase "Max Steps" from 500 to 1000 and change "DX_MAX" from .3 to .1 for better electron trajectories. The screen now appears as in figure 5.12. Exit this form and select "Make Script" to write out file "tr_traj_in." Exit "DynaPre."

Now, at the UNIX prompt, issue the command:

```
%Tracker_SUN4 < tr_plot_in > tr_plot_out & |
```

Then run your favorite "fort.2 plot-reader" to display the plot as in figure 5.13. On color plot, ions are shown in red and electrons are shown in green. Note that particles in bipolar zones are not generated.

You might want to save this fort.2 file as it will be overwritten by the next command. At the UNIX prompt, type the command

```
%Tracker_SUN4 < tr_traj_in > tr_traj_out &
```

Again, run your favorite "fort.2 plot_reader" to display the particle trajectories as in figure 5.14.

Note that ion particles got sucked right into the object while electrons were just oscillating about the magnetic field until they fell into high-field region between the two cubes where they were pulled into the cube's face.

Some new subfiles have been created by PartGen and Tracker here:

2Cubes.PT1	Contains particle attributes, created by PartGen and updated by Tracker as particles are tracked.
2Cubes.POTG	Contains time-dependent spatial potentials.
2Cubes.POTS	Contains time-dependent surface potentials.
2Cubes.CUR	Contains time-dependent current to surface information.

It is interesting to look at trajectories of fewer electron particles instead. Now, rerun "PartGen" with command:

```
%PartGen_SUN4 < pg_in > pg_out
```

Although you could run "DynaPre" again to build a new "tr_traj_in", it is simple enough to make the change manually using a text editor. Edit the following lines in "tr_traj_in" to read:

```
X_LIMIT  7.0  7.5
Y_LIMIT  7.0 13.0
Z_LIMIT  0.0  0.0
```

Then, run "Tracker" again with command:

```
%Tracker_SUN4 < tr_traj_in > tr_traj_out
```

Once more, run your favorite plotter to display the new trajectories, which will be as in figure 5.15

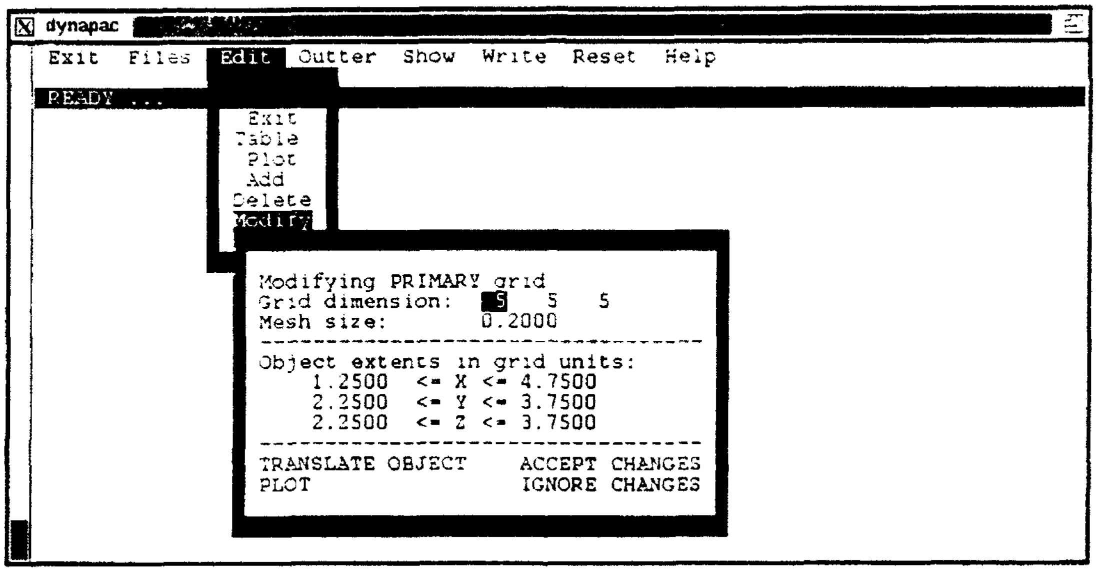
Now, just for fun, let's change the magnetic field to be along X-axis and track sheath particles on the X=8 plane and within 5.5 < Y < 6, 5 < Z < 6 range. Run "DynaPre" to build appropriate "pg_in" and "tr_traj_in" input files to "PartGen" and "Tracker," respectively. Then run "PartGen" and "Tracker", *e.g.*

```
%PartGen_SUN4 < pg_in > pg_out
%Tracker_SUN4 < tr_traj_in > tr_traj_out
```

Finally, plot the trajectories, which will look like figure 5.16.

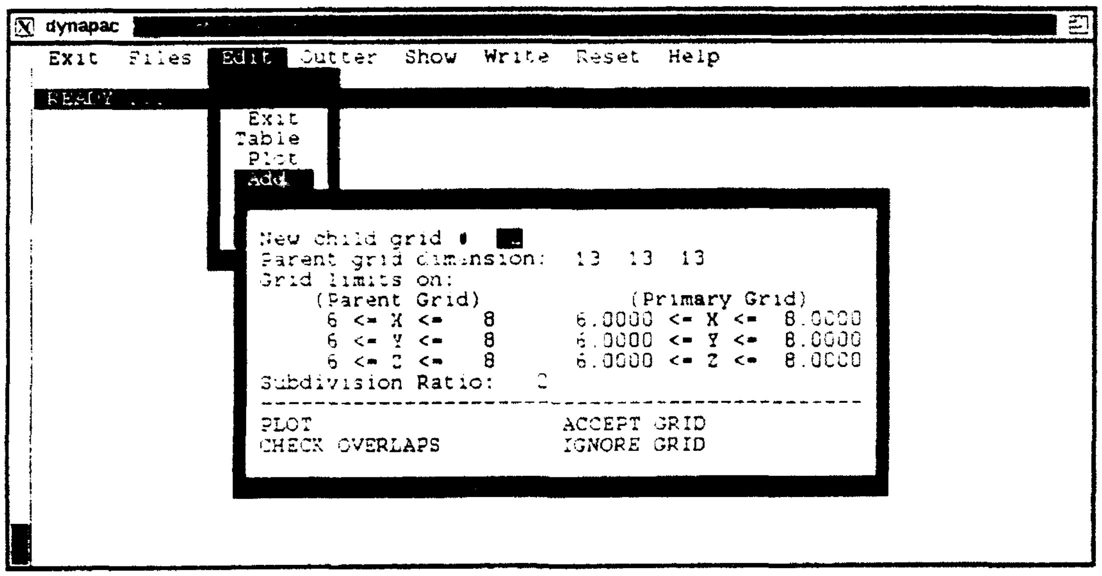
5.11 Summary

This example has taken you through most of DynaPAC modules: defining an object; setting up computational grids; calculating and displaying potentials in the surrounding space; generating, tracking particles, and displaying particle's trajectories.



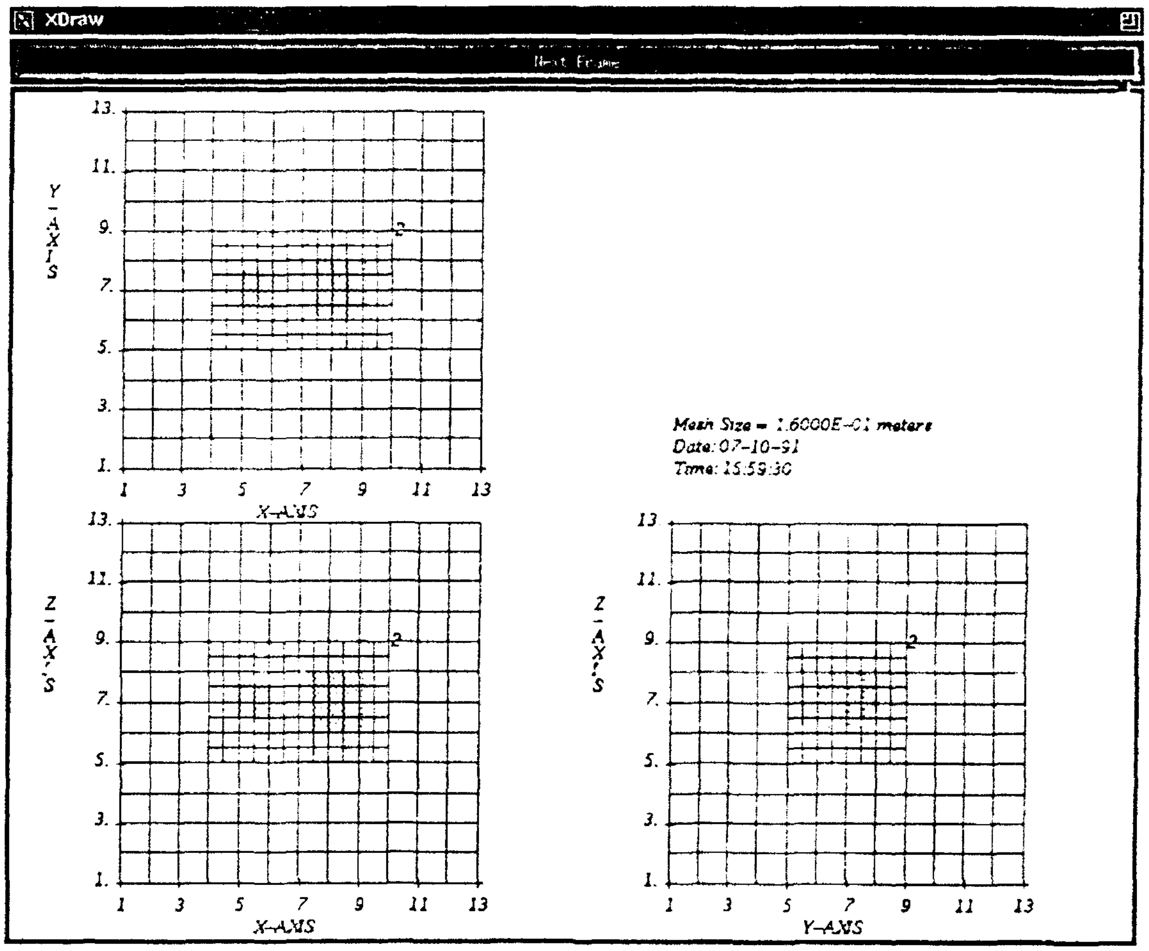
```
Modifying PRIMARY grid
Grid dimension:  5    5    5
Mesh size:       0.2000
-----------------------------------
Object extents in grid units:
    1.2500  <= X <= 4.7500
    2.2500  <= Y <= 3.7500
    2.2500  <= Z <= 3.7500
-----------------------------------
TRANSLATE OBJECT     ACCEPT CHANGES
PLOT                 IGNORE CHANGES
```

Figure 5.1 GridTool screen for the Edit/Modify option.



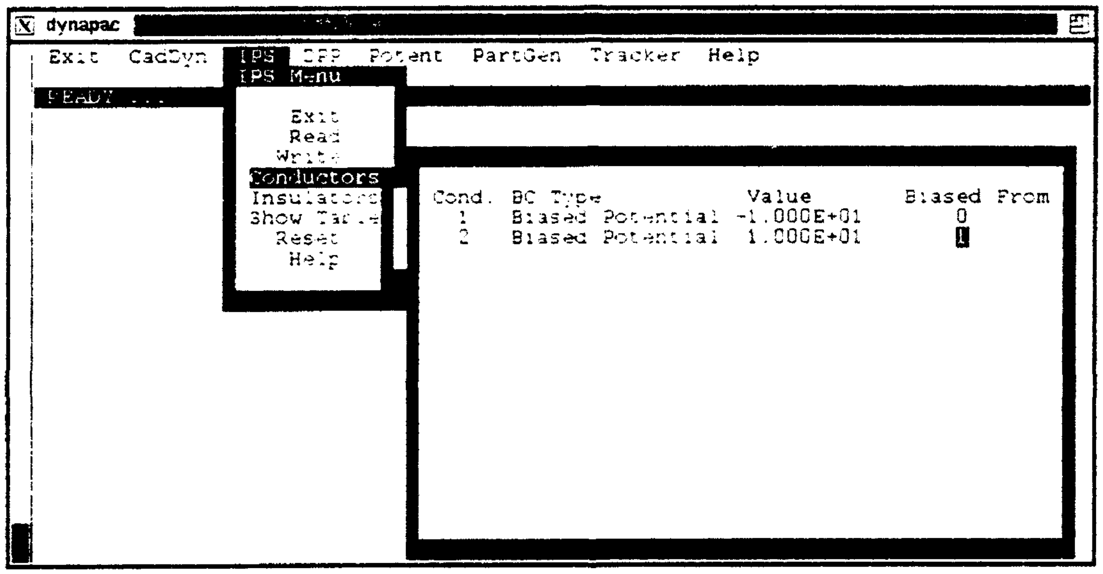
```
New child grid #
Parent grid dimension:  13  13  13
Grid limits on:
     (Parent Grid)             (Primary Grid)
      6 <= X <=    8       6.0000 <= X <=  8.0000
      6 <= Y <=    8       6.0000 <= Y <=  8.0000
      6 <= Z <=    8       6.0000 <= Z <=  8.0000
Subdivision Ratio:    2
-----------------------------------------------------
PLOT                          ACCEPT GRID
CHECK OVERLAPS                IGNORE GRID
```

Figure 5.2 GridTool screen for the Edit/Add option.



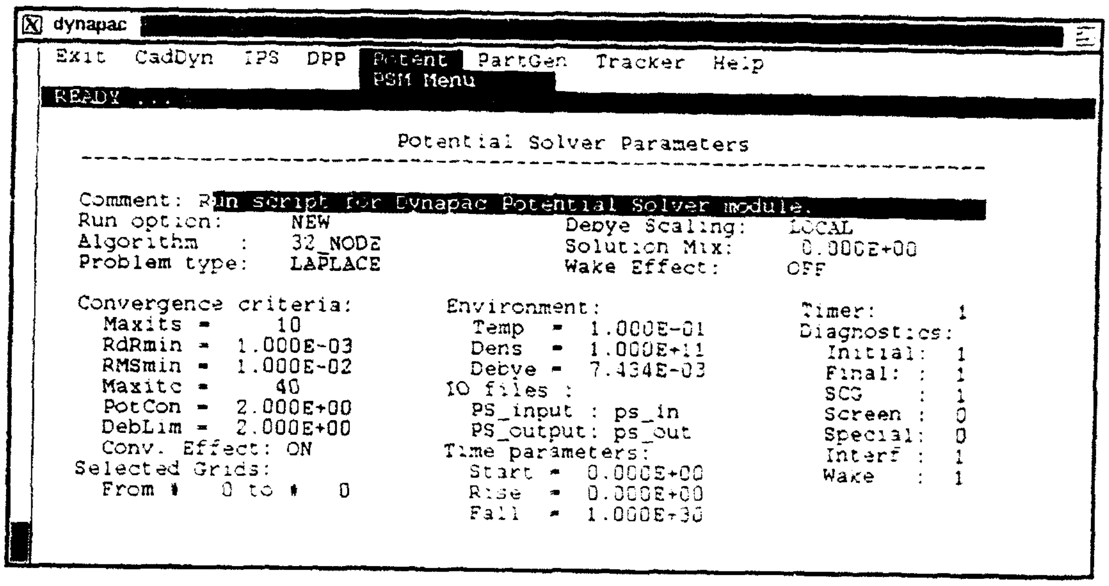
```
Mesh Size = 1.6000E-01 meters
Date: 07-10-91
Time: 15:59:30
```

Figure 5.3 GridTool plot for the 2-Cubes sample problem.



```
Cond.  BC Type                Value       Biased From
  1    Biased Potential  -1.000E+01           0
  2    Biased Potential   1.000E+01           1
```

Figure 5.4 DynaPre screen for the IPS/Conductors option.

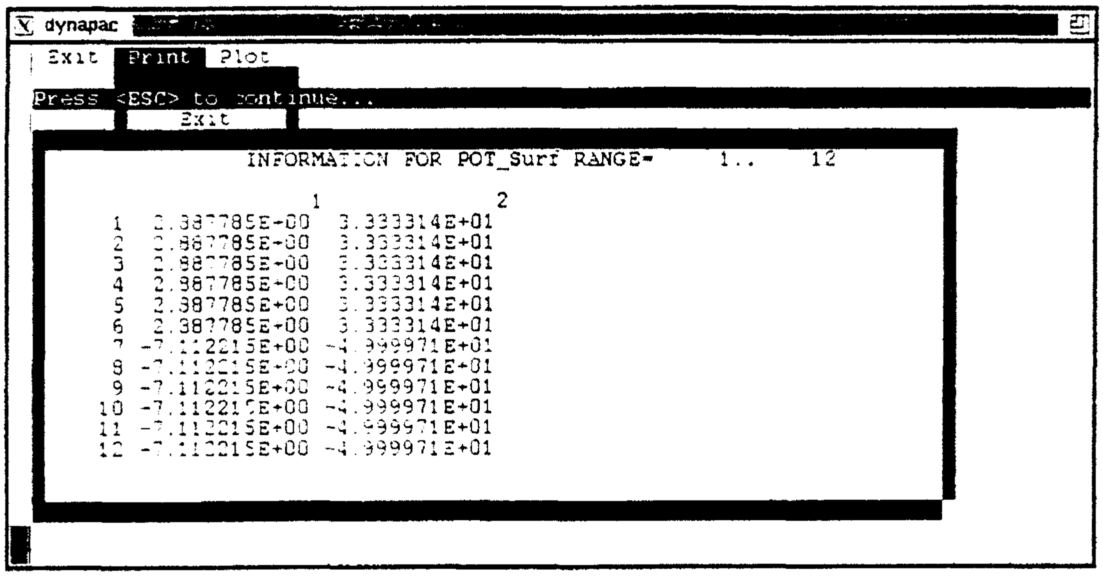


```
                          Potential Solver Parameters
 -------------------------------------------------------------------

 Comment: Run script for Dynapac Potential Solver module.
 Run option:      NEW                         Debye Scaling:    LOCAL
 Algorithm    :   32_NODE                     Solution Mix:      0.000E+00
 Problem type:    LAPLACE                     Wake Effect:      OFF

 Convergence criteria:       Environment:                 Timer:        1
   Maxits  =      10           Temp   =  1.000E-01        Diagnostics:
   RdRmin  =  1.000E-03        Dens   =  1.000E+11          Initial:  1
   RMSmin  =  1.000E-02        Debye  =  7.434E-03          Final:  : 1
   Maxitc  =      40         IO files :                     SCG     : 1
   PotCon  =  2.000E+00        PS_input  : ps_in            Screen  : 0
   DebLim  =  2.000E+00        PS_output: ps_out            Special:  0
   Conv. Effect: ON          Time parameters:               Interf  : 1
 Selected Grids:               Start  =  0.000E+00          Wake    : 1
   From #   0 to #   0         Rise   =  0.000E+00
                               Fall   =  1.000E+30
```

Figure 5.5 DynaPre screen for the POTENT/Edit Script option.

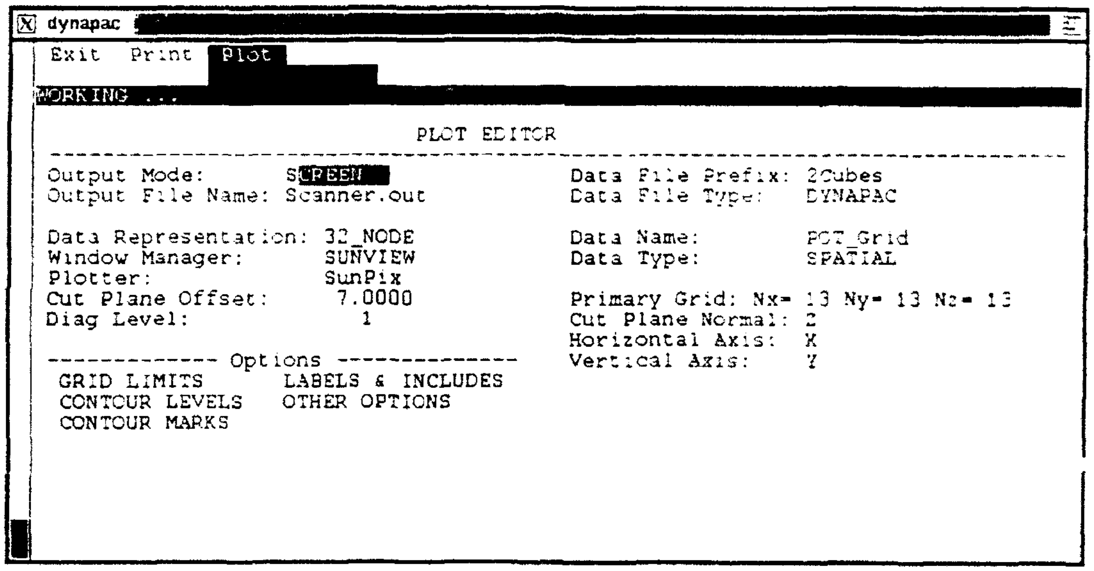


```
Press <ESC> to continue...

                INFORMATION FOR POT_Surf RANGE=     1..     12

                      1              2
   1   2.387785E+00   3.333314E+01
   2   2.387785E+00   3.333314E+01
   3   2.387785E+00   3.333314E+01
   4   2.387785E+00   3.333314E+01
   5   2.387785E+00   3.333314E+01
   6   2.387785E+00   3.333314E+01
   7  -7.112215E+00  -4.999971E+01
   8  -7.112215E+00  -4.999971E+01
   9  -7.112215E+00  -4.999971E+01
  10  -7.112215E+00  -4.999971E+01
  11  -7.112215E+00  -4.999971E+01
  12  -7.112215E+00  -4.999971E+01
```

Figure 5.6 Surface Floating Potentials and Fields.



```
dynapac
 Exit   Print   Plot
WORKING ...

                               PLOT EDITOR
-------------------------------------------------------------------------------
Output Mode:       SCREEN                  Data File Prefix: 2Cubes
Output File Name:  Scanner.out             Data File Type:   DYNAPAC

Data Representation: 32_NODE               Data Name:        PCT_Grid
Window Manager:      SUNVIEW               Data Type:        SPATIAL
Plotter:             SunPix
Cut Plane Offset:     7.0000               Primary Grid: Nx= 13 Ny= 13 Nz= 13
Diag Level:             1                  Cut Plane Normal: Z
                                           Horizontal Axis:  X
------------- Options --------------       Vertical Axis:    Y
 GRID LIMITS        LABELS & INCLUDES
 CONTOUR LEVELS     OTHER OPTIONS
 CONTOUR MARKS
```

Figure 5.7 Scanner screen for the Plot/Edit Plot option.

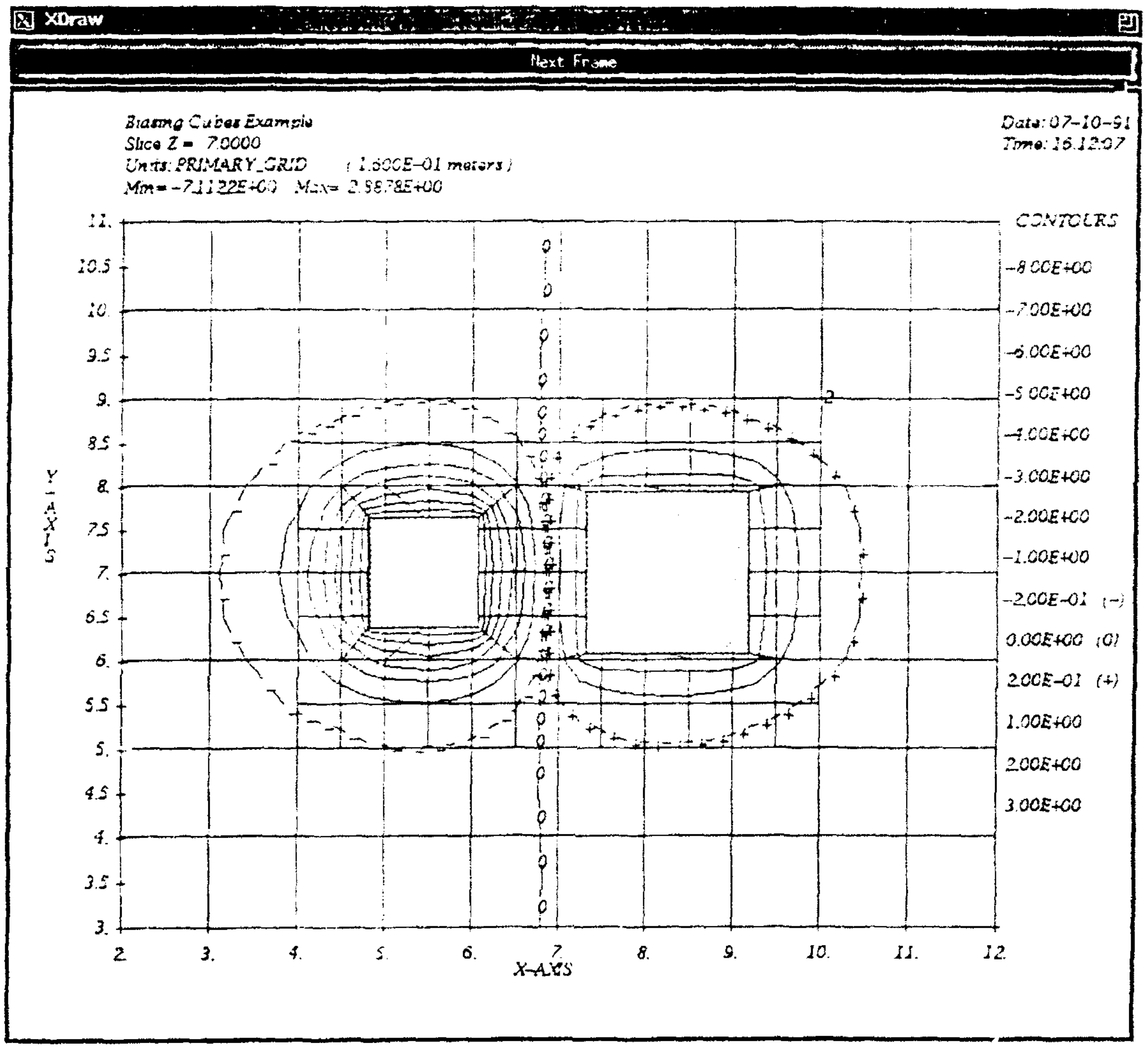
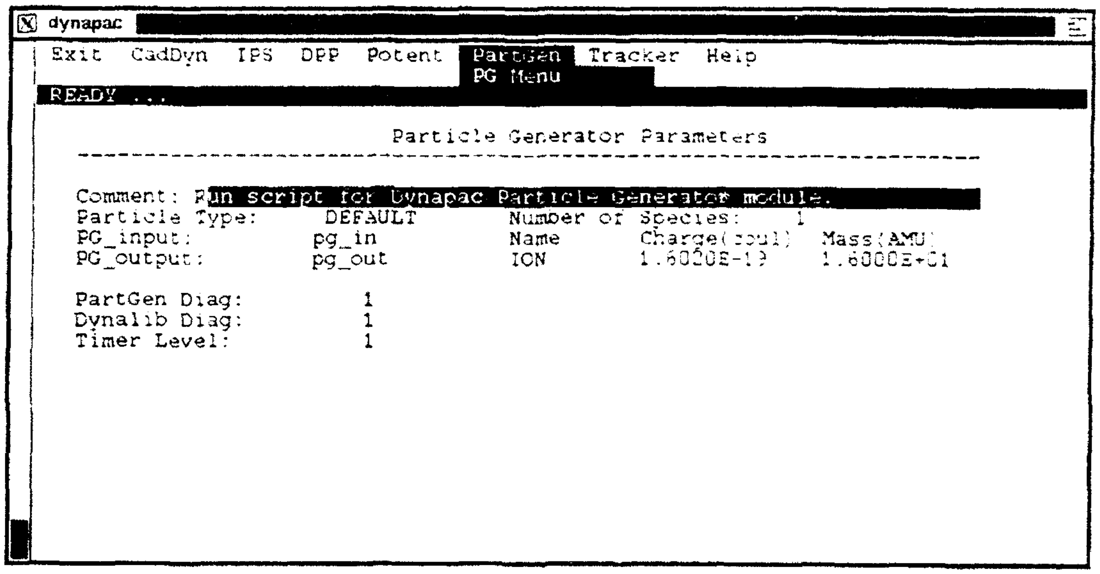


Figure 5.8 Contour plot of potentials for sample problem, black-and-white version.

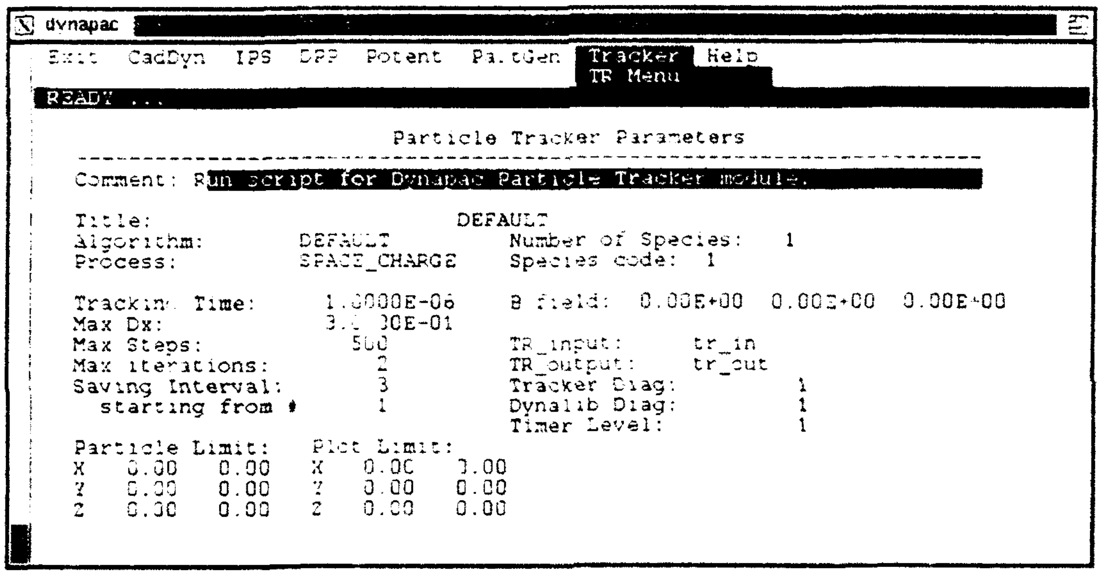


```
                    Particle Generator Parameters
 ----------------------------------------------------------------

 Comment: Run script for Dynapac Particle Generator module.
 Particle Type:      DEFAULT        Number of Species:    1
 PG_input:          pg_in           Name      Charge(coul)   Mass(AMU)
 PG_output:         pg_out          ION       1.6020E-19     1.6000E+01

 PartGen Diag:         1
 Dynalib Diag:         1
 Timer Level:          1
```

Figure 5.9 DynaPre/PartGen editing screen.

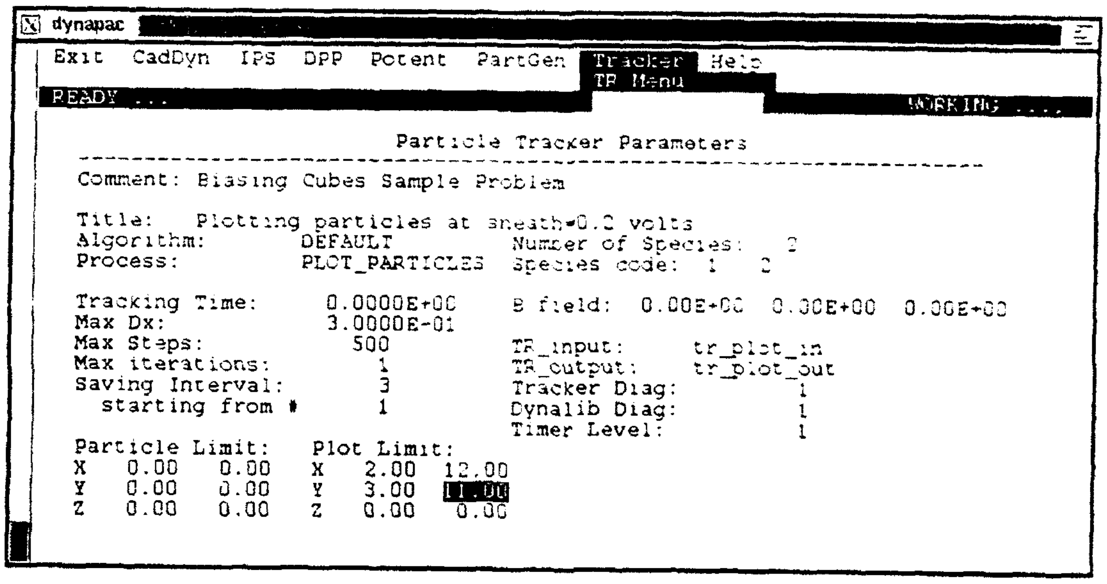


```
                     Particle Tracker Parameters
 ---------------------------------------------------------------
 Comment: Run script for Dynapac Particle Tracker module.

 Title:                          DEFAULT
 Algorithm:          DEFAULT          Number of Species:    1
 Process:            SPACE_CHARGE     Species code:  1

 Tracking Time:       1.0000E-06      B field:  0.00E+00  0.00E+00  0.00E+00
 Max Dx:              3.0 00E-01
 Max Steps:             500           TR_input:       tr_in
 Max iterations:          2           TR_output:      tr_out
 Saving Interval:         3           Tracker Diag:          1
   starting from #        1           Dynalib Diag:          1
                                      Timer Level:           1
 Particle Limit:    Plot Limit:
 X   0.00    0.00   X   0.00    0.00
 Y   0.00    0.00   Y   0.00    0.00
 Z   0.00    0.00   Z   0.00    0.00
```

Figure 5.10 DynaPre/Tracker editing screen.

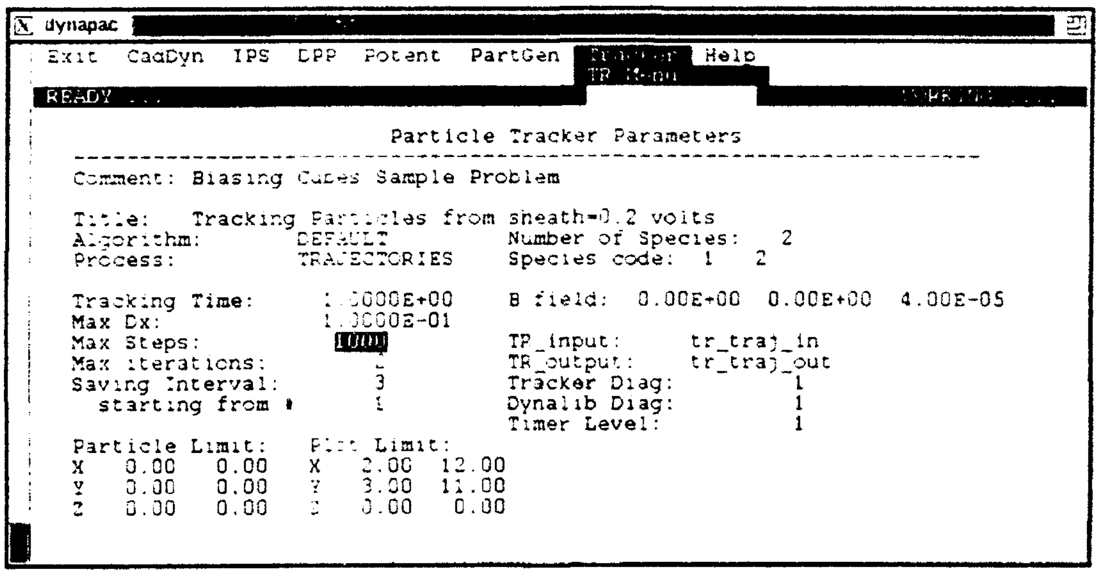


```
                    Particle Tracker Parameters
 ---------------------------------------------------------------------
 Comment: Biasing Cubes Sample Problem

 Title:   Plotting particles at sheath=0.2 volts
 Algorithm:          DEFAULT          Number of Species:   2
 Process:            PLOT_PARTICLES   Species code:  1   2

 Tracking Time:       0.0000E+00      B field:  0.00E+00  0.00E+00  0.00E+00
 Max Dx:              3.0000E-01
 Max Steps:              500          TR_input:      tr_plot_in
 Max iterations:           1          TR_output:     tr_plot_out
 Saving Interval:          3          Tracker Diag:          1
   starting from #         1          Dynalib Diag:          1
                                      Timer Level:           1
 Particle Limit:    Plot Limit:
 X    0.00     0.00    X    2.00   12.00
 Y    0.00     0.00    Y    3.00   11.00
 Z    0.00     0.00    Z    0.00    0.00
```

Figure 5.11 DynaPre/Tracker editing PLOT_PARTICLES screen.

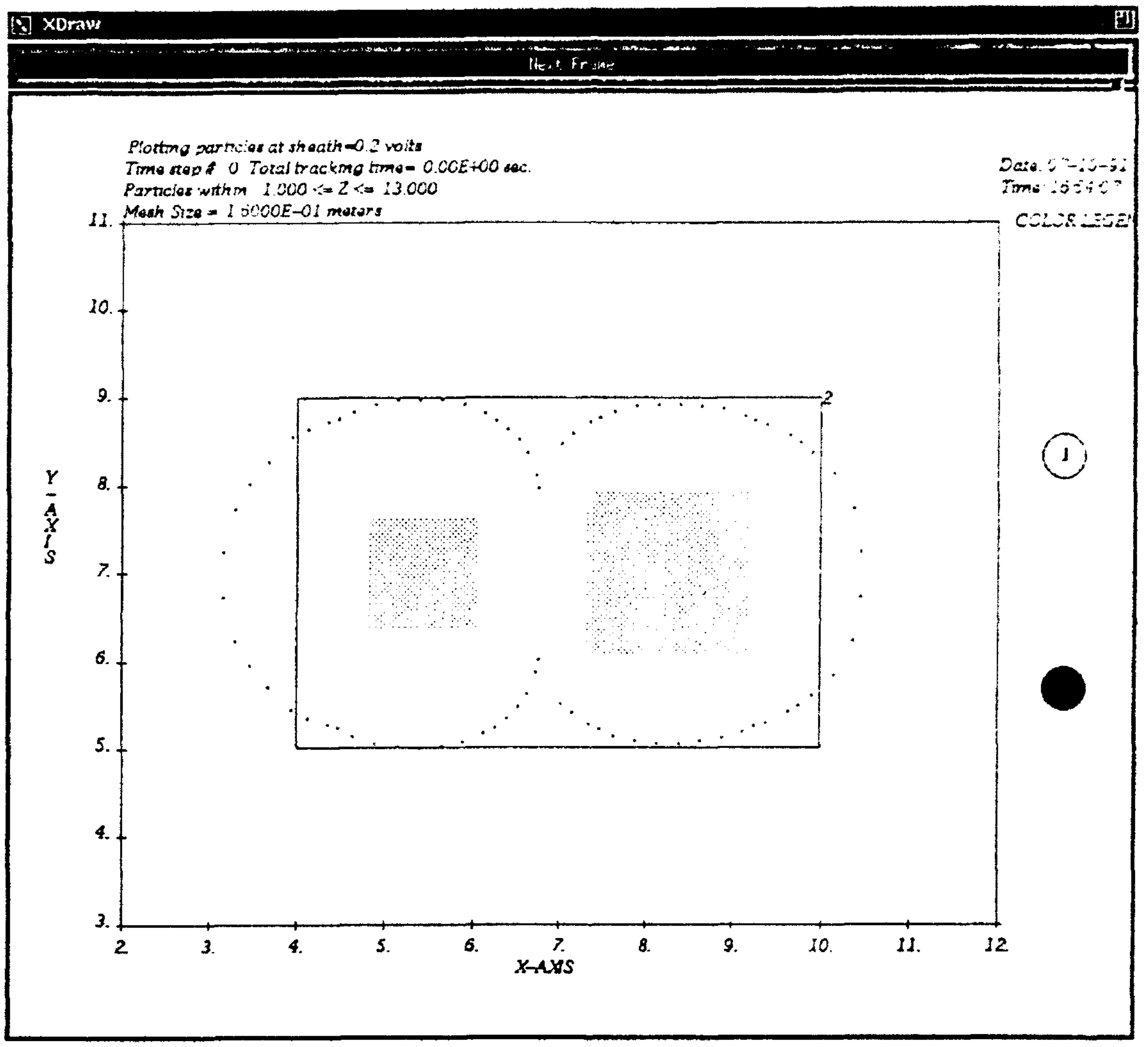


```
                     Particle Tracker Parameters
 -------------------------------------------------------------------
 Comment: Biasing Cubes Sample Problem

 Title:    Tracking Particles from sheath=0.2 volts
 Algorithm:        DEFAULT          Number of Species:   2
 Process:          TRAJECTORIES     Species code:  1   2

 Tracking Time:     1.0000E+00      B field:  0.00E+00  0.00E+00  4.00E-05
 Max Dx:            1.0000E-01
 Max Steps:           1000          TR_input:      tr_traj_in
 Max iterations:         1          TR_output:     tr_traj_out
 Saving Interval:        3          Tracker Diag:          1
   starting from #       1          Dynalib Diag:          1
                                    Timer Level:           1
 Particle Limit:    Plot Limit:
 X   0.00    0.00   X   2.00  12.00
 Y   0.00    0.00   Y   3.00  11.00
 Z   0.00    0.00   Z   0.00   0.00
```

Figure 5.12 DynaPre/Tracker editing TRAJECTORIES screen.



Plotting particles at sheath=0.2 volts
Time step # 0 Total tracking time= 0.00E+00 sec.
Particles within 1.000 <= Z <= 13.000
Mesh Size = 1.6000E-01 meters

Figure 5.13 Particle at sheath=0.2 volts.

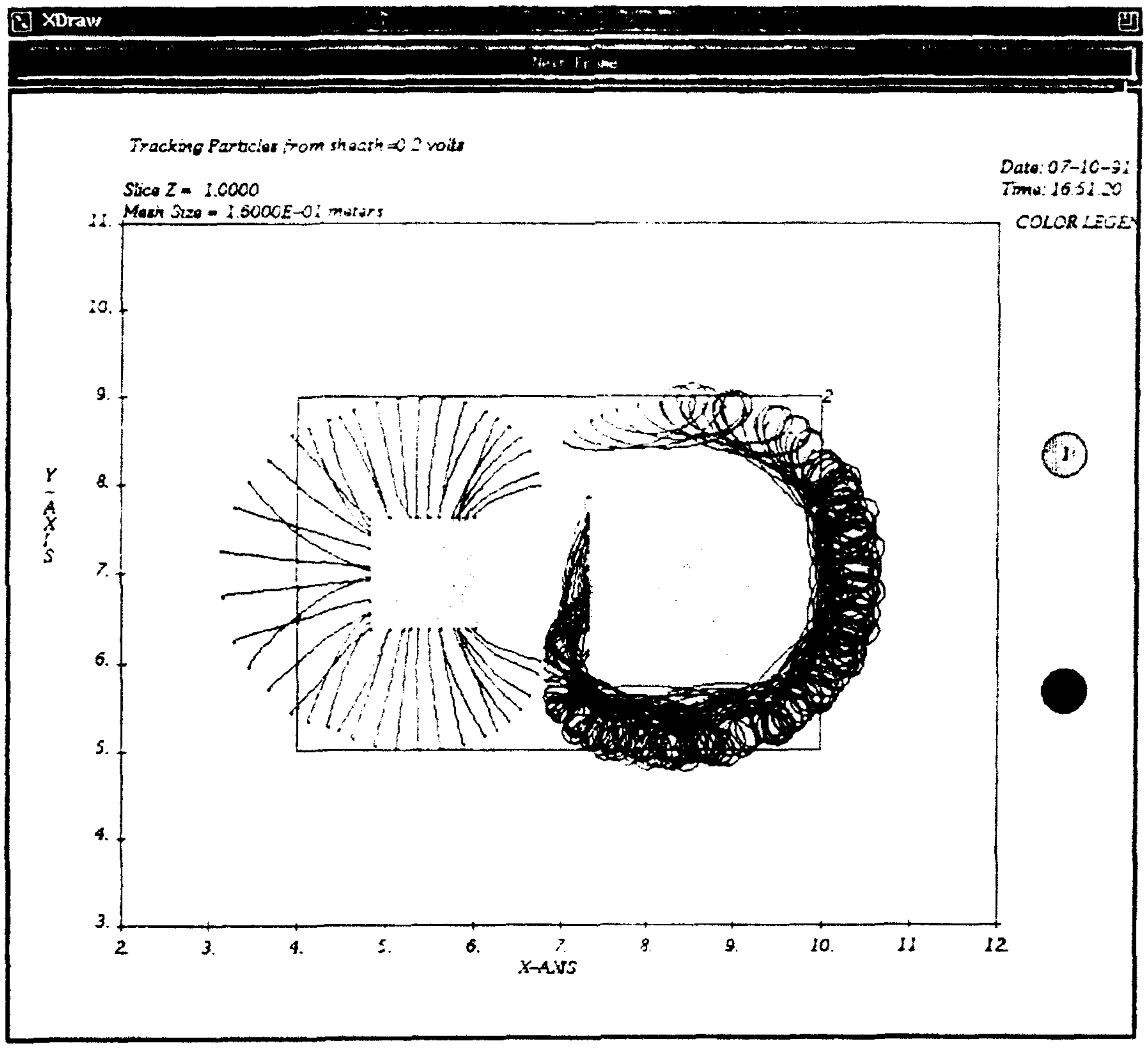


Figure 5.14 Particle trajectories.

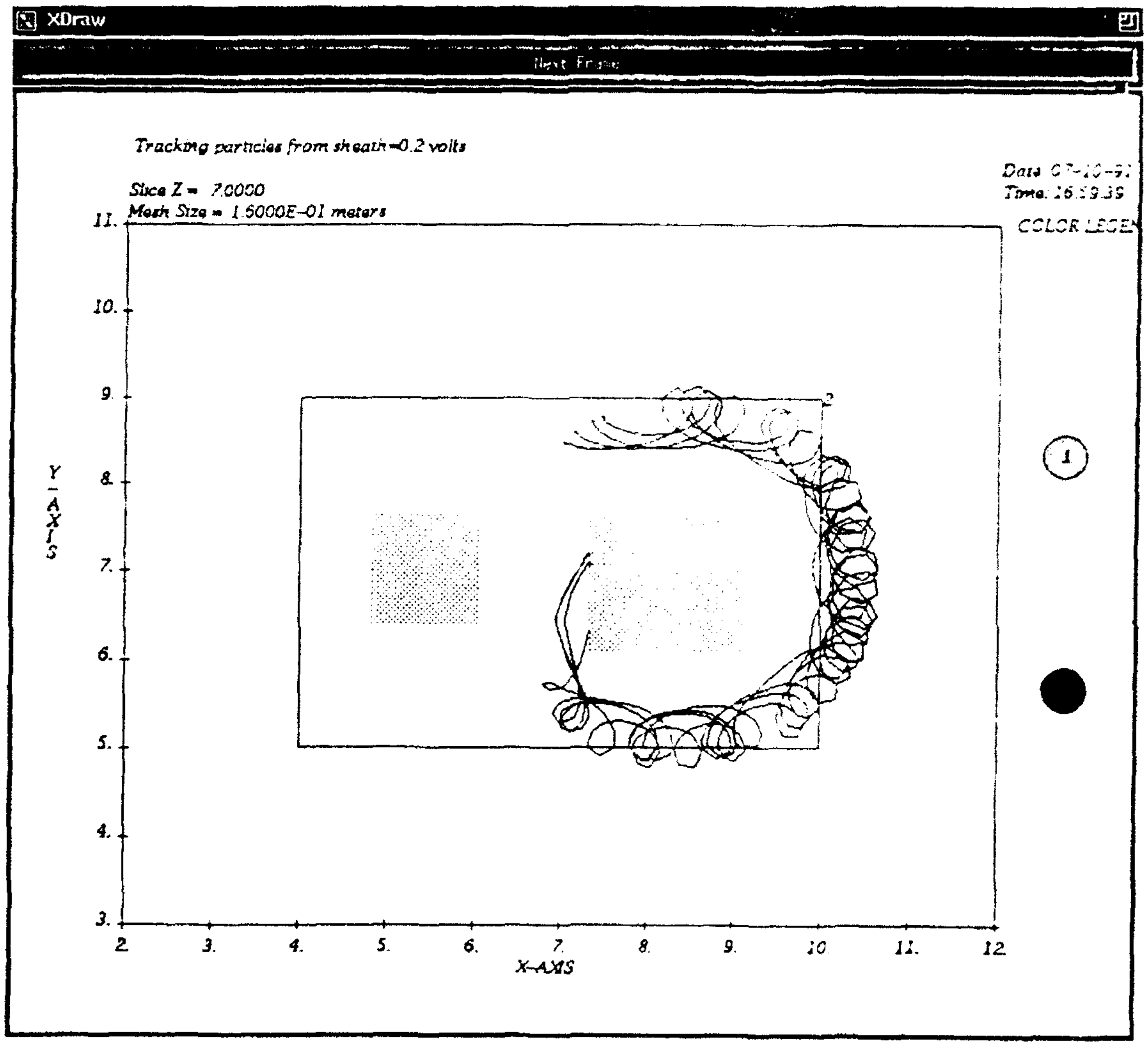


Figure 5.15 Electron trajectories.

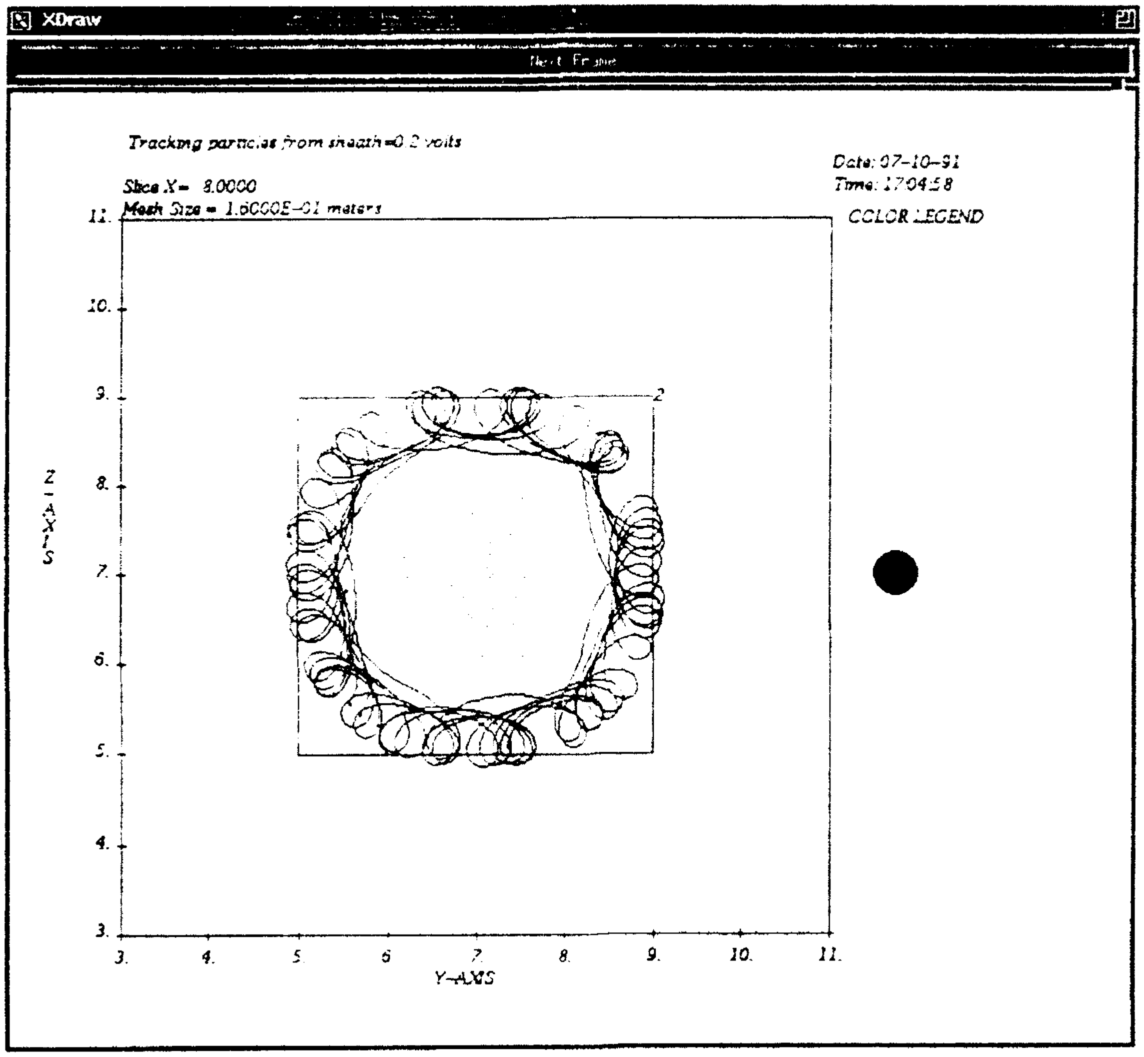


Figure 5.16 Electron trajectories 2.

6. Documentation for Two-D Electrostatic Code "Gilbert"

6.1. Introduction

Gilbert is a general, two-dimensional (R-Z or X-Y) finite-element electrostatic code. It can solve a fairly general version of Poisson's equation, and can emit and track charged particles in electric and magnetic fields.

Poisson's equation is solved taking into account varying dielectric constants in different regions. Boundary conditions include fixed potential conductors, floating conductors, and internal "windowscreen" boundary conditions. Space charge is contributed by charged particles being tracked, by charged particles deposited on insulating surfaces, by conductivity effects, and by analytic "plasma" formulas.

Charged particles may be normally emitted from conducting surfaces at fixed energy and current density. Zero energy particles will be emitted in space-charge-limited fashion. An applied magnetic field in the Z-direction (for either geometry) may be specified. Up to five different particle species may be specified. Initial particles may be generated using the auxiliary routine *InitP*.

All information concerning gridding, particle properties, input variables, potential values, etc. is maintained in a single database (stored as fort.20). Auxiliary programs to make plots or to otherwise query the database are included in the *Gilbert* package. Additional special-purpose programs may easily be written by users.

Presently *Gilbert* is implemented under UNIX on a Ridge 3200 (space). However, it should be readily convertible to any other UNIX machine, or to VAX/VMS. Graphical programs are at present designed for PostScript printers, although they can also be displayed on Tektronix 4014 terminals or emulators. Color output is anticipated.

6.2. Grid Generation

6.2.1. Gridding Requirements

Gilbert operates on a finite-element grid composed of an arbitrary combination of linear triangles, bilinear (4-node) quadrilaterals, and biquadratic (8-node) quadrilaterals. (In general, the 8-node quads will give superior results.) These polygons are referred to as "elements", and their corners (or mid-side points for the 8-node quads) are referred to as "nodes". Each element has associated with it an integer property ($1<M<15$) called a material number. Material 1 is considered free-space, and supports propagation of charged particles. Materials 2-15 are considered dielectrics, and are capable of charge deposition on their surfaces. Each node has associated with it an integer property ($0<B<20$) called a "boundary condition". Nodes with boundary condition 0 are free-space nodes. Nodes with boundary conditions 1-20 are conductor nodes, with user-specified properties (see below). Free-space nodes on the boundary of the grid will have zero-normal-electric-field boundary conditions.

An external mesh generator (such as Patran) is used to generate the grid. The mesh generator must provide a sequential list of nodes, with their locations and boundary conditions, and a sequential list of elements, with their defining nodes and material numbers. Elements must be compatible, i.e., a side of a polygon (or half-side of an 8-node quad) must also be a side of another polygon or it will be considered a boundary. (Note that in a triangular region some mesh generators will supply 6-node triangles where 8-node quads have been requested. Such elements are not currently acceptable to *Gilbert*, and must be rezoned by the user.) The mesh generator should be used to eliminate unintended internal boundaries. Also, the mesh generator should be used to minimize abrupt changes in grid spacing, and to eliminate particularly badly misshapen elements.

6.2.2. Grid Generation using Patran

Pending further request, the only available mesh generator interface, *ReadPat*, reads the Patran "neutral file". Patran (a product of PDA Engineering) is available at S-CUBED on the Silicon Graphics Iris, and may be run with graphical interaction on the Iris console, or in "BAT" mode from any other terminal. We also have software capable of converting an IGES file to a Patran neutral file. Consult Myron Mandell or Victoria Davis concerning access to Patran or Patran documentation.

For *Gilbert* we use Patran to construct a two-dimensional mesh in the X-Y plane (i.e., the default plane of the screen). The nodes may later be scaled (to account for units), and interpreted as X-Y, Y-X, R-Z, or Z-R. Element material numbers correspond the Patran MIDs, and node conductor numbers correspond to the Patran node configuration number.

It is not our intent here to provide a Patran tutorial, but we will give a brief description of a few of the concepts and commands which might be used to generate a grid.

(Brief Patran Discussion to be inserted)

It is not necessary to assure that all elements are counterclockwise, as this operation will be performed by *ReadPat*.

6.2.3. Grid Processing with *ReadPat*

ReadPat creates and initializes the database (fort.20) with the gridding information and with default values for various other options and parameters. *ReadPat* expects to find the Patran neutral file on logical unit 8 (fort.8). It also accepts certain options from standard input, and writes an output file on standard output.

The options accepted by *ReadPat* on standard input are:

Input Syntax	Meaning	Default
DPRN	Diagnostic Print	Omit Diagnostic Print
ECHO	Echo Patran Neutral File	No Echo
NOEC	Do not echo Patran Neutral File	No Echo
FLIP	Interpret input as Y-X or Z-R	Interpret input as X-Y or R-Z
UNIT r	Input unit is r meters	r = 1.
UNIT INCH	Input unit is .0254 meters	
UNIT FEET	Input unit is .3048 meters	
UNIT FOOT	Input unit is .3048 meters	
UNIT CENT	Input unit is .0100 meters	
UNIT METE	Input unit is 1 meter	
UNIT MILL	Input unit is .0010 meters	
DIEL i r	Material i has dielectric constant r	Dielectric constant = 1.
SIGM i r	Material i has conductivity r	Conductivity = 0.
RZ	(No Effect)	Geometry is R-Z unless Rmin < 0
XY	Geometry is X-Y	
END	No further input	

6.2.4. Grid Illustration with *GridPlot*

After creating the database with ReadPat, it is possible to produce PostScript plots of the interpreted grid using the *GridPlot* program. *GridPlot* will read the database (fort.20) and create graphics commands on logical unit 2 (fort.2). *PostPlot* will translate these commands into PostScript commands on its standard output, which may be piped or otherwise sent to a convenient PostScript printer. Alternatively, *Tek4014* will write Tektronix 4014 commands on its standard output.

GridPlot will draw element outlines and node points, and, for R-Z geometry, will draw the axis of symmetry. Elements of materials 2-15 will be shaded in gray. At user preference, *GridPlot* can print element numbers, element material numbers, conductor numbers, or node numbers. To accept a choice, respond to the prompt with a simple carriage return <CR>. Any non-null response will reject the choice.

GridPlot has a windowing capability, allowing plots to be restricted to only a portion of the grid.

6.3. Running *Gilbert*

6.3.1. General Considerations

Gilbert is run in the directory where its database (fort.20) resides. It reads an input file from standard input, writes output on standard output, and continually updates its database. Calculation takes place in four phases: (1) read input and related initialization; (2) generate Laplace equation matrices (if necessary); (3) initialize potentials (if necessary); (4) perform iterative cycles to resolve non-linear and/or time-dependent phenomena.

Gilbert first reads in all current values of options and parameters from the database (fort.20). It is envisioned that a screen-oriented input program will be made available to manipulate these options and parameters. At present, however, user-modifiable options and parameters may be changed via command from standard input. These options and parameters and their input syntax are described in the next section.

After inserting the mesh into the database with *ReadPat*, it is necessary to generate the matrices to be used to solve for the potential, as well as to perform certain other initializations. This is done in response to the "GenMat" command. ("GenMat" is set to .True. by *ReadPat*. It is automatically set to .False. after the initialization is performed.)

Potentials in space, and charges on floating conductors, are initialized in response to the command "IniPot". The variable "IniPot" may be set to .True. from the input stream. It will be set to .False. following the initialization.

The variable "Cycle" determines whether iterative cycles are to be performed. It is set to .True. when a non-zero number of cycles is requested via the input variable "NCYC". Each cycle consists of four phases: (1) emit particles, track the particles, and deposit charge accordingly on conductors and insulating surfaces; (2) computed charge accumulation due to conductivity effects; (3) calculate space charge due to particles, particle deposition, and plasma effects, and update the potential solution accordingly; (4) generate written and printed output as requested. If further cycles are required, subsequent runs may resume from the current state of the database.

6.3.2. Common Block/Input Specifications

6.3.2.1. Boundary Conditions

COMMON/BCOND/BCTyp(NCMX),VOLTS(NCMX), BCParm(NCMX), Capaci(3,NCMX)

NCMX is a parameter denoting the maximum number of conductors (currently 20). BCTyp describes the type of boundary condition. Currently implemented values are "FIX " for fixed potential conductors, "FLOQ" for charge-initialized floating conductors, "FLOV" for potential-initialized floating conductors, "IGNO" to indicate that this collection of nodes is not to be treated as a conductor, and "SCRE" for windowscreen internal boundary. VOLTS describes the voltage of a fixed-potential boundary, the initial voltage of a potential-initialized floating boundary, or the nominal voltage of a windowscreen boundary. BCParm may contain either the parameter relating effective voltage to field discontinuity for a windowscreen boundary, or the initial charge for a charge-initialized floating boundary. Capaci is an array containing interconductor capacitances. The first two entries of each triplet are integers giving the conductor numbers to be capacitively connected, and the third is the capacitance in farads (R-Z Geometry) or farads/meter (X-Y Geometry). Note that changes in capacitance values will be implemented only during the matrix generation (GenMat) phase of the code.

Relevant input variables are:

Input Syntax	Meaning	Default
FIX i	Fixed potential on conductor i	Floating Potential
VOLT i volts	Define potential on conductor i	Volts(I) = 0.
FLOA i	Potential-initialized floating potential on conductor i	(Default)
IGNO i	Ignore specification for conductor i	
CHAR i coul	Charge initialized floating conductor i, with coul coulombs of charge	Voltage-initialized
SCRE i alpha	Windowscreen internal boundary on conductor i, with parameter alpha	Voltage-initialized
CAPA ic1 ic2 farads	Interconductor Capacitance	Stray Capac. only
CAPA RESET	zero all capacitances	

6.3.2.2. Plasma Parameters

```
Common /Plasma/ Rho, T0, RLamda, Dsq, Linear, Planar, Convrg,Elec,Ions
Logical Linear, Planar, Convrg,Elec,Ions
```

This common block contains the plasma density, temperature, and Debye length. Dsq is the square of the Debye length. The three logical variables describe the formulation to be used for plasma space charge, with the options "Linear" for linear (Debye) screening, "Planar" for Child-Langmuir screening, "Convrg" for spherically convergent

(Langmuir-Blodgett) screening, "Elec" for screening by only the electron component of the plasma (*e.g.*, if ions are tracked), and "Ions" for screening by only the ion component of the plasma (*e.g.*, if electrons are tracked). Relevant input variables are:

Input Syntax	Meaning	Default
PLAS LINE	Linear Screening	No Screening or As Before
PLAS PLAN	Planar Screening	No Screening or As Before
PLAS CONV	Convergent Screening	No Screening or As Before
PLAS ELEC	Electron Screening	No Screening or As Before
PLAS IONS	Ion Screening	No Screening or As Before
PLAS other	Turn off Screening	As Before
DEBY length	Set Debye Length; Redefine RHO	7430 meters, or As Before
RHO density	Set RHO; Redefine RLamda	1 m^{-3}, or As Before
TEMP temp	Set T0; Redefine RLamda	1 eV, or As Before

6.3.2.3. Mesh Parameters

```
COMMON/MESH/NNodes,NElt,IBSet,RMin,RMax,ZMin,ZMax,RZGeom,XYGeom
Logical RZGeom,XYGeom
```

NNodes and NElt describe the number of nodes (maximum 3100) and elements (maximum 3000) actually in the grid. RMin, RMax, ZMin, and ZMax describe the minimum and maximum R (or X) and Z (or Y) coordinates of the nodes. The logical variables RZGeom and XYGeom describe whether the geometry is to be treated as R-Z or X-Y. (Note that this common block contains substantial further gridding information, which is omitted here.)

Relevant input keywords are "XY" and "RZ", which reset the geometry type. (Note that if the geometry type is actually reset, the matrix initialization ("GenMat") should be redone.)

6.3.2.4. Particle Parameters, Magnetic Field, Timestep

```
COMMON /PARCON/ EQ(5), EM(5), EByM(5), BZ, Omega(5), DelT, Mirror
Logical Mirror
```

EQ and EM give the (signed) particle charge (coulombs) and mass (kilograms) for the species of particles of each species to be tracked. EByM = EQ/EM. BZ is the z-component of magnetic field (W-m^{-2}). Omega = EByM*BZ. DelT is the timestep per cycle, and also the maximum timestep per particle push. Mirror indicates that Z=0 (for R-Z) or Y=0 (for X-Y) is a mirror plane. Relevant input variables (applied to the current particle species number) are:

Input Syntax	Meaning	Default
PART SPEC n	Particle Species for subsequent input	1
PART CHAR coul	Particle charge (coulombs)	-1.6×10^{-19}
PART MASS kg	Particle mass (kilograms)	9.1×10^{-31} kg
PART MASS a AMU	Particle mass (amu)	9.1×10^{-31} kg
BZ w	Magnetic field (W-m^{-2})	0.
BZ g GAUS	Magnetic field (gauss)	0.
DELT sec	Timestep (seconds)	$1.\times10^{-9}$
MIRROR	Declare Mirror Plane	No Mirror
MIRROR OFF	Omit Mirror Plane	No Mirror

6.3.2.5. Emission parameters

```
Common /EMISSN/ Window(4),CurDen(NCMX),Energy(NCMX),NPSEG(NCMX),
   *             JSpeces(NCMX),NPMod(NCMX)
```

Window give the XMin, XMax, YMin, YMax from which emission can take place. (This is useful if only a portion of a conductor is emitting.) CurDen gives the maximum current density (A-m^{-2}) emitted from each conductor. Energy gives the energy at which particles are emitted, with zero energy indicating space-charge-limited emission. NPSeg gives the number of particles to be emitted for each line-segment of conductor. JSpeces gives the species number of particles to be omitted from each conductor, as specified by the most recent "PART SPEC n" card. (See particle specifications section above.) NPMod gives the number of timesteps between emissions. (Useful for problems having both slow and fast particles.)
Relevant input parameters are:

Input Syntax	Meaning	Default
EMIS WIND x1 x2 y1 y2	Define emission window	Entire Grid
EMIT COND i ENER e CURD j NPSE n Mod m	Define emission properties for conductor i	(No emission)

6.3.2.6. Calculation Directives

```
Common /WhToDo/ GenMat, IniPot, Cycle,LstCyc, NCyc, Time
Logical GenMat, IniPot, Cycle
```

GenMat defines whether Laplace matrix initialization is to be performed. IniPot defines whether potential initialization is to be performed. Cycle defines whether iterative cycles are to be performed. LstCyc defines the number of the previous cycle or timestep. NCyc defines the number of cycles to be performed. Time is the running total of the DelT's for each timestep. Input parameters are:

Input Syntax	Meaning	Default
NCYC n	Number of cycles; Set CYCLE to .True.	No cycles
GENM	Initialize Laplace matrix	Set to .True by *ReadPat* Set to .False. after done
INIP	Initialize Potentials	.False.
CYCLE RESET	Set LstCyc=Time=0 Maintain existing particles and charge deposition.	Continue from previous

6.3.2.7. Particle Tracking Parameters

Common /Track/ XYNoB, XYB,RZNoB, RZB,DXMax,DYMax,DEMax
Logical XYNoB,XYB,RZNoB,RZB

The four logical variables are set internally to determine the appropriate type of particle pushing. DXMax and DYMax define the maximum X (or R) and Y (or Z) distance a particle can move in a timestep. If this is exceeded with timestep DelT, then multiple particle timesteps will be taken within the cycle for that particle. Similarly, DEMax is the maximum kinetic energy change per particle push. Relevant input variables are:

Input Syntax	Meaning	Default
DXMA dx	Maximum DX per particle push	$0.1\times$(RMax-RMin)
DYMA dy	Maximum DY per particle push	$0.1\times$(ZMax-ZMin)
DEMA de	Maximum kinetic energy change per particle push	1×10^6 eV

6.3.2.8. Other Miscellaneous parameters

Common /MISC/ MaxItr, IPrPar, ParPsh, Conduc
Logical ParPsh, Conduc

MaxItr is the maximum number of iterations within the ICCG potential solver. IPrPar turns on particle diagnostics. ParPsh defines whether any particle pushing is required. (ParPsh is set to .True. when emission is requested, but may also be set by the user.) Conduc defines whether conductivity is to be considered. (Conduc is initially set to .False. by *ReadPat*, and changed to .True. when *ReadPat* encounters a 'SIGM' input card assigning conductivity to a material. Relevant input variables are:

Input Syntax	Meaning	Default

MAXI n	Maximum number of ICCG iterations	30
IPRP n	Particle Diagnostics	0
PARP	Turn on particle pushing	
PARP OFF	Turn off particle pushing	
COND	Turn on conductivity	
COND OFF	Turn off conductivity	

6.3.2.9. Output Specifications

It is commonly desired to print or plot a quantity as a function of time or cycle number. When Subroutine Input encounters an 'OUTP' card, it transfers control to Subroutine OutSpc to interpret the remainder of the card. The capabilities and syntaxes of this feature are discussed below in the section titled "Output Discussion".

6.3.3. Windowscreen Boundary Condition

We have incorporated into the code an algorithm for treating the windowscreen as an interior boundary condition. The method determines the local effective potential of the windowscreen self-consistently with the calculated electric fields on either side of the screen.

Normally, the windowscreen spacing is smaller than any other parameter (resolution or Debye Length) in the problem. In this case, the difference between the effective potential of the windowscreen and its actual (circuit) potential is proportional to the discontinuity of electric field across the screen:

$$V_{eff} - V_s = -a(T)\, L\, (\mathbf{E}_a - \mathbf{E}_b) \bullet \mathbf{n}$$

where L is the interwire spacing, a(T) is a function of the transparency of the screen, $\mathbf{E}_{a,b}$ are the electric fields above and below the screen, and **n** is the upward unit normal to the screen. The parameter ("alpha") which must be entered when specifying the "SCRE" boundary condition is a(T)×L. The table below gives values of a(T) for various transparencies.

Transparency	a(T)
50%	.0185
60%	.0310
70%	.0488
80%	.0754
90%	.120
95%	.174

6.3.4. User-Modifiable Routines

It is not possible to calculate in a general way every variation on the electrostatic theme. However, we have tried to anticipate some of the areas in which users will wish to "hardwire" parameters and functional forms for their particular applications, and to localize these in user-modifiable routines. The routines intended for such purposes are extensively internally documented. We list these routines below:

6.3.4.1. Subroutine ExtCur

ExtCur provides the capability of imposing a time-dependent potential on "FIX " and "SCRE" conductors, or providing an external current (which may depend on time and potential) to floating conductors.

6.3.4.2. Subroutine Output

Output provides the capability of printing or writing for later processing several categories of quantities in a general, automated way. However, the user may have specific, unanticipated quantities of which he would like to keep track. A stencil is provided in Subroutine Output for the user to calculate and print/write such a quantity.

6.3.5. Output Discussion

6.3.5.1. User Requested Output

The user may request output to be printed or to be written on unformatted file fort.30. The output may be conductor or node charges, conductor or node potentials, nodal fields, and total particle charge or dipole moment. A program *LinPlt* is provided to plot the data on fort.30. As with other computational parameters, the output requests are stored in common blocks and maintained in the database for restart purposes. We will, as done with input parameters above, categorize the output requests by common block.

Changes in output requests should be done only when the INIP request is used to establish the calculation at cycle 1.

6.3.5.1.1. General Requests

Common /OMods/ ModPrt, ModWrt

The cards

```
OUTPUT MODP nprint
OUTPUT MODW nwrite
```

specify that printing/writing takes place every nprint/nwrite cycles. A value of zero indicates no printing/writing. The default is to set both values to zero. Also, the card

OUTPUT RESET

sets the default values for all output parameters **except** ModPn and ModWn.

6.3.5.1.2. Conductor-Related Quantities

```
Common /OCnd/ IConds(20), LChrge, LPot, LEmit, LRetrn
Logical LChrge, LPot, LEmit, LRetrn
```

The values in the above common block are controlled by the cards

```
OUTPUT CONDUCTOR icond [OFF]
OUTPUT CONDUCTOR CHARGE [OFF]
OUTPUT CONDUCTOR POTENTIAL [OFF]
OUTPUT CONDUCTOR EMISSION [OFF]
OUTPUT CONDUCTOR RETURN [OFF]
```

The default is to print/write potential, not to print/write charge, cumulative emission current, or cumulative return current, and to have no conductors specified.

6.3.5.1.3. Nodal Potentials

```
Common /OPots/ NodPot(20)
```

Nodal potentials are requested by the card

```
OUTPUT POTENTIAL NODE inode [OFF]
```

By default, no nodes are specified.

6.3.5.1.4. Nodal Fields

```
Common /OFlds/ NodFld(2,20), LE1, LE2, LETot
Logical LE1, LE2, LETot
```

Nodal fields are requested by the cards

```
OUTPUT FIELD NODE inode [ELEMENT ielt]
OUTPUT FIELD NODE inode [OFF]
OUTPUT FIELD TOTAL [OFF]
OUTPUT FIELD dir [OFF]
```

where dir is one of [R, X, 1] to print/write the R or X field component, and one of [Z, Y, 2] to print/write the Z or Y field component. If no element is specified for a node, *Gilbert* will determine an appropriate element. The default is to have no nodes or elements specified, and to print/write only the total electric field.

6.3.5.1.5. Particle Quantities

```
Common /OParts/ LQPart(6), LDip1(6), LDip2(6)
Logical LQPart, LDip1, LDip2
```

Particle quantities are requested by the cards

```
OUTPUT PARTICLE SPECIES n
OUTPUT PARTICLE CHARGE [OFF]
OUTPUT PARTICLE DIPOLE dir [OFF]
```

where a value n from 1 to 5 specifies a particle species for subsequent OUTPUT PARTICLE commands (any other value indicating sum over all particles), and dir is one of [R, X, 1] to print/write the R or X dipole component, and one of [Z, Y, 2] to print/write the Z or Y dipole component. The default is to omit calculation of particle quantities.

6.4. Auxiliary Programs

6.4.1. GridPlot

GridPlot has been described above.

6.4.2. InitP

InitP allows the user to specify initial particles throughout the grid. The user will be prompted for a density for each particle species. If a positive density is given, InitP will place one macroparticle on each triangle or 4-node quadrilateral, and four macroparticles on each 8-node quad. *InitP* should logically be run following an initial potential (INIP) calculation by *Gilbert*.

6.4.3. Contours

Contours allows the user to plot equipotential contours in the grid. Contour values may be either quasi-logarithmically spaced (1, 2, 5, 10, 20, ...) or evenly spaced. Windowing capability allows plotting to be limited to a portion of the grid.

6.4.4. Scatter

Scatter plots the positions of particles of each species currently in the problem.

6.4.5. LinPlt

LinPlt allows the user to plot quantities written to file fort.30 via the output options. The quantities may be plotted either vs. time or vs. cycle number, and a range of cycles may be selected. Also, the user may select the quantities to be plotted.

The user may readily create custom versions of *LinPlt* to plot multiple curves on a plot or to plot derived quantities.